DATA TRANSLATION®

UM-16150-C

DT VPI™ User's Manual

Version 6.0

Third Edition
July, 2000

Copyright © 1997, 1998, 2000 by Data Translation, Inc.
Copyright © 2000 by Agilent Technologies.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording, or otherwise, without the prior written permission of Data Translation, Inc.

Information furnished by Data Translation, Inc. is believed to be accurate and reliable; however, no responsibility is assumed by Data Translation, Inc. for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent rights of Data Translation, Inc.

Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer software clause at 48 C.F.R, 252.227-7013, or in subparagraph (c)(2) of the Commercial computer Software - Registered Rights clause at 48 C.F.R., 52-227-19 as applicable. Data Translation, Inc., 100 Locke Drive, Marlboro, MA 01752.

Data Translation® is a registered trademark, and DT VPI™ is a trademark of Data Translation, Inc.

All other brand and product names are trademarks or registered trademarks of their respective companies.

Data Translation, Inc.
100 Locke Drive
Marlboro, MA 01752-1192
(508) 481-3700

www.datatranslation.com
Fax: (508) 481-8620
E-mail: info@datx.com

Table of Contents

About this Manual xi
What is DT VPI? xi
Overview of Data Acquisition xi
What Can My Board Do? xiv
Intended Audience xv
What You'll Need xv
What You Should Learn from this Manual xvi

Chapter 1: Installing DT VPI 1
Step 1: Install Your Data Acquisition Board 3
Connecting the Screw Terminal Panel 3
Connecting a Function Generator for Input 4
Connecting an Oscilloscope for Output 4
Step 2: Install the Device Driver 4
Step 3: Install VEE 4
Step 4: Install DT VPI 5

Chapter 2: Quick Start 7
The Basics 8
Starting VEE 8
Getting Help 11
Starting VEE Online Help 11
Starting DT VPI Online Help 11
Quitting VEE 13
Working with Objects 14
Adding an Object to the Work Area 14
Moving an Object 15
Selecting the Object Menu 15

Copying an Object 16
Cutting an Object 17
Pasting an Object 18
Changing Object Views 18
Changing the Size of an Object 19
Changing the Title of an Object 20
Understanding Pins and Terminals 22
Connecting Objects 23
Adding Data Input Terminals to an Object 23
Deleting Data Input or Output Terminals from an Object 24
Renaming a Terminal 24
Understanding Propagation 25

Chapter 3: Your First Program 29
Lesson 1: Create A Simple Program 31
Step 1: Add the Objects 31
Step 2: Connect the Objects 32
Step 3: Run Your Program 33
Step 4: Change Object Parameters 34
Step 5: Print the Screen 36
Step 6: Save Your Program 36
Lesson 2: Modify Your Simple Program 38
Step 1: Make Some Noise 38
Step 2: Add an Amplitude Input 40
Step 3: Run! 41
Lesson 3: Acquire Analog Data 43
Step 1: Gather the Equipment 43
Step 2: Add the Objects 44
Step 3: Configure the A/D Subsystem 45
Step 4: Run! 47

Step 5: Acquire Continuously 48
Step 6: Save Your Program 49
Lesson 4: Acquire with Get Single Value 50
Step 1: Gather the Equipment 50
Step 2: Add the Objects 51
Step 3: Configure the A/D Subsystem. 52
Step 4: Run! 53
Step 5: Save Your Program 54
Step 6: Collect the Data. 55
Step 7: Build a Waveform. 56
Step 8: Save and Run! 58
Lesson 5: Output Analog Data 60
Step 1: Gather the Equipment 61
Step 2: Add the Objects 61
Step 3: Configure the D/A Subsystem. 62
Step 4: Run! 64
Step 5: Save Your Program 65
Lesson 6: Digital Input and Output. 66
Step 1: Gather the Equipment 66
Step 2: Add the Objects 67
Step 3: Configure the Digital Input Port 68
Step 4: Run! 69
Step 5: Save Your Program 69
Lesson 7: Change the Configuration Programmatically. 70
Step 1: Gather the Equipment 70
Step 2: Modify the Program. 71
Step 3: Add the Configuration Objects 72
Step 4: Add User Input. 74
Step 5: Run! 76
Step 6: Save Your Program 77

What's Next? 78

Chapter 4: Analog Input 79
Connecting External Sensors 81
Configuring for Analog Input 81
Selecting a Board when More than One Data Acquisition Board Is Installed. 82
Building a Channel List 83
Selecting a Sampling Rate 84
Filtering an Analog Input Signal 85
Selecting the Gain 86
Selecting an Input Range and Type 87
Selecting a Clock and Trigger Source. 88
Specifying a Subsystem Handle 89
Starting an Analog Input Acquisition 91
Acquiring Data from a Single Channel. 92
Acquiring Data from Multiple Channels 94
Acquiring Analog Data Continuously 96
What's Really Going On?. 98
States of an A/D Channel 100
Acquiring a Single Analog Value. 101
Acquiring a Burst of Data 104
Configuring for Burst Mode 105
Using an External Trigger 107
What's Really Going On?. 108
Acquiring Pre-Trigger Data 109
Configuring for Pre-Trigger Acquisition. 110
Acquiring and Retrieving Pre-Trigger and Post-Trigger Data 111
Acquiring Data Using a Triggered Scan 112
Changing the Subsystem Configuration. 117

Handling Acquisition Errors 121
Errors When Writing to File 122
Trapping Errors 122
Sending Data to a Disk File. 125
Writing to File Using the To File Object 125
Writing Directly to File. 130
Configuring Each Channel for Writing to a File. 133
Configuring Multiple Channels for Writing to a Single File 135
Configuring for Slow Sampling Rates 138
Sending Data Through DDE. 139
Optimizing Analog Input 142

Chapter 5: Analog Output 145
Connecting External Devices 147
Configuring for Analog Output 147
Selecting a Board when More than One Data Acquisition Board Is Installed. 148
Configuring the Channel List 149
Selecting an Output Rate 150
Selecting an Output Range and Type. 153
Specifying a Subsystem Handle 154
Starting an Analog Output 155
Outputting Analog Data from a Single Channel. 156
Outputting Analog Data Continuously 159
What's Really Going On? 160
States of a D/A Channel 162
The Critical Threshold 163
Outputting a Single Analog Value. 164

Outputting a Burst of Data 166
Configuring for Burst Mode 166
What's Really Going On? 169
Changing the Subsystem Configuration 170
Handling Acquisition Errors 175
Errors When Reading From File 175
Trapping Errors 176
Reading Data From a Disk File 177
Receiving Data From DDE 180
What's Really Going On? 182
Optimizing Analog Output 183

Chapter 6: Digital Input and Output 185
Connecting External Devices 187
Configuring for Digital I/O 187
Acquiring a Single Digital Value 189
Outputting a Single Digital Value 191

Chapter 7: Simultaneous A/D and D/A 193
Hardware Support 195
Starting Two Subsystems Simultaneously 200
Sharing DMA Resources 201

Chapter 8: Using Counters and Timers 203
Terminology 205
Pulse Type 205
Duty Cycle 205
Gate Types 206
Delay and Pulse Width 207
Duration 208
Clock Frequency 208

Using the Counter/Timer as a Clock 209
Configuring the C/T as a Clock 210
What's Really Going On? 211
Using the Counter/Timer as a Trigger 212
Configuring the C/T as a Trigger 213
What's Really Going On? 214
Taking it Further: An External Trigger to the C/T 215
Using the Counter/Timer to Count Events 216
Configuring the C/T to Count Events 216
What's Really Going On? 217
Using the Counter/Timer to Measure Frequency 218
Configuring the C/T to Measure Frequency 219
What's Really Going On? 220
Optimizing Counter/Timers 221

Chapter 9: Developing Programs for VEE RunTime 223

Chapter 10: Advanced Functionality 227
cdtv.SetResolution 237
dtv.GetSingleValueEx 238
dtv.PutSingleValueEx 238
dtv.GetTriggerThreshold 239
dtv.SetTriggerThreshold 239
Accessing the Unsecured Data Acquisition UserObjects 240
Thermocouple Support 241
ThAverage 242
CompensationVolts 242
VoltsToDegrees 243

Appendix A: Example Programs 245

Appendix B: Error Codes 251
DT VPI Errors 252
DT-Open Layers Errors 260

Appendix C: File Formats 267
Single Channel Format 268
Multi-Channel Format 269
Channel-Gain List Entries 271

Appendix D: Product Support 273
General Checklist 274
Service and Support 275
Telephone Technical Support 275
E-Mail and Fax Support 278
World-Wide Web 278
Glossary 279

Index 295

About this Manual

What is DT VPI?

Agilent Technologies® and Data Translation® formed an alliance to provide the broadest range of solutions for data acquisition and test and measurement under Microsoft® Windows®. Agilent VEE™ Visual Programming Language solves the performance needs of high channel count and instrument-based I/O.

The alliance enabled DataTranslation to create a component called DT VPI to support data acquisition from within the VEE visual programming environment. DT VPI™ is targeted at PC-based data acquisition requiring low point count (typically under 100 channels). The name DT VPI stands for Data Translation's Visual Programming Interface. It provides you with a visual interface consisting of VEE objects specifically designed to control data acquisition hardware from Data Translation. Together, DT VPI and VEE offer a spectrum of solutions for many applications.

This manual describes how to use DT VPI with VEE.

Overview of Data Acquisition

When you think of "data acquisition," you typically think of acquiring analog data from some sort of external sensor or measuring device, such as a strain gauge, thermocouple, microphone, or pressure transducer. The data you are interested in could be a single value, a series of single values you want to obtain at timed intervals, or a continuous stream of data. In the latter case the data may be coming in too fast to acquire all of it, so you would like to take a representative sample of the data, a snapshot in time, if you will. Using an onboard analog-to-digital converter (ADC), your data acquisition board obtains an accurate representation, or *sample*, of

data. Every so often, hundreds or thousands of times a second, a value of analog data is recorded. The frequency at which a value is recorded is called the *sampling rate* (expressed in samples per second), or *sampling frequency* (expressed in hertz). The sampling process is called *analog-to-digital conversion*, or A / D conversion.

To perform multiple conversions automatically at precisely-defined time intervals, most data acquisition boards from DataTranslation are equipped with one or more *pacer clocks*. A pacer clock typically starts all conversions. Boards that permit simultaneous analog input (A / D) and analog output (D / A) operations often have separate A / D and D / A pacer clocks.

Pacer clocks are made up of a *frequency source* (either an onboard free-running oscillator or a user-supplied signal) and a divider network that steps the frequency source down to more usable values. The frequency source is also called the clock's *base frequency*, and determines the granularity or distance between available settings. Higher base frequencies offer finer granularity, especially at high sampling frequencies. For example, values for a 400 kHz oscillator are 2.5 µs apart; those for a 5 MHz oscillator are only 0.2 µs apart. Dividers typically can be set for any value within an 8- or 16-bit range. Some designs also use a binary prescaler before the divider to extend the clock's range.

Pacer clocks are started by *triggers*: either by a program instruction (a software trigger), or by an external digital pulse or analog voltage at the board's connector (an external trigger). External clock frequency sources and external triggers help you synchronize data conversions with offboard events; an external trigger can serve as a gate to enable conversions, while with each tick, the external frequency source starts a new conversion.

Using DT VPI and your plug-in data acquisition board, analog or digital data can be acquired through external sensors connected to your board. Your data acquisition board has one or more *subsystems* on it, which work independently to acquire or output analog or digital data. The following subsystems are provided:

- Analog input (A / D);
- Analog output (D / A);
- Digital input (DIN);
- Digital output (DOUT); and
- Counter / timer (C / T).

DT VPI provides objects that interact with the subsystems on your data acquisition board. These objects acquire, analyze, display, or output data. For each subsystem, there is an object used to configure the subsystem before it begins to operate. During configuration, your data acquisition board is informed about which channels are going to be used for data acquisition or output, and at what sampling rate and gain. This information is passed to every other data acquisition object that needs to interact with that subsystem on the data acquisition hardware.

In the following chapters, each subsystem is discussed in detail. You will learn how to make external connections, configure the subsystem, and to connect and use related data acquisition objects. Whenever possible, the examples on the distribution disks are used and explained. In addition, the intricacies of using two or more systems simultaneously, as well as optimizing your data acquisition program, are described.

What Can My Board Do?

You can write data acquisition programs in VEE with DT VPI using any data acquisition board from DataTranslation. This manual describes how to use the data acquisition objects provided by DT VPI to write programs to acquire and output analog and digital data. With the exception of Chapter 7, this manual does not mention particular boards.

Some boards support features that other data acquisition boards do not. For example, some have 16-bit resolution, and some have 12-bit resolution. Some have hardware anti-aliasing filters (such as DT3831) and some do not. If you need assistance in determining what your board supports, refer to the manuals that describe your board and device driver.

Determine what capabilities your board supports by checking the supported options table in your board's device driver documentation. For example, to determine if your board supports continuous data acquisition with gap free data, check the row labelled "Continuous Operation Support." If there is "Yes" in the column under the A / D subsystem, then your board can continuously acquire gap free data.

Depending on the capabilities of your particular board, some features may be grayed in the configuration dialog boxes. A grayed feature indicates that the feature is unavailable for the selected data acquisition board.

Intended Audience

This manual is written for engineers and scientists who have some experience with programming and working with instruments. It assumes some knowledge of Microsoft Windows. It is assumed you have VEE Pro 6.0 or VEE OneLab 6.0 from Agilent Technologies plus DT VPI from Data Translation, and a data acquisition board from Data Translation.

What You'll Need

To use DT VPI, you need the following:

- Microsoft Windows 95, Windows 98, Windows NT® 4.0, or Windows 2000.
- Pentium 120 MHz or faster processor.
- 20 MB minimum RAM for Windows 95 and Windows 98; 48 MB minimum RAM for Windows NT 4.0 or Windows 2000.
- A monitor that supports a minimum of SVGA (800-by-600).
- A CD-ROM drive.
- A hard disk drive with at least 2.7 MB of free disk space for data acquisition and example programs. This space is in addition to the space required to install VEE.
- VEE Pro 6.0, VEE Pro Runtime 6.0, or VEE OneLab 6.0.
- An installed data acquisition board from DataTranslation.
- An input and / or output device connected to the data acquisition board.

What You Should Learn from this Manual

This manual describes how to install DT VPI "on top of" VEE Pro (development or runtime), or VEE OneLab, and how to use DT VPI's data acquisition objects to control your data acquisition hardware from Data Translation. The data acquisition objects look and behave just like other VEE objects. After reading this manual, you should understand how to use the data acquisition objects in your VEE programs to acquire, process, and output analog and digital I/O data.

This manual does *not* cover other aspects of VEE programming, such as the flow of data, using the Math objects, and creating UserObjects. To learn about these aspects of programming in VEE, you need to read through the user manuals that are supplied with VEE.

The manual is organized as follows:

- Chapter 1, "Installing DT VPI," describes how to install DT VPI, and refers you to your data acquisition hardware manual if you have not yet installed your hardware.
- Chapter 2, "Quick Start," covers the basics of using objects. If you are already familiar with VEE, then you can skip this chapter.
- Chapter 3, "Your First Program," guides you through several lessons on using the data acquisition objects in DT VPI.
- Chapter 4, "Analog Input," describes how to configure the A/D subsystem and create programs to acquire analog data. Many different types of analog input are discussed, such as continuous input, burst mode, single value, as well as sending data to disk and through DDE.
- Chapter 5, "Analog Output," describes how to configure the D/A subsystem and create programs to output analog data. Continuous, burst mode, and single value output are discussed, as well as how to set up the Function Generator object to generate data for output.

- Chapter 6, "Digital Input and Output," explains how to configure the digital input and output subsystems, and how to acquire and output single digital values.
- Chapter 7, "Simultaneous A/D and D/A," describes what you need to know to run two subsystems simultaneously.
- Chapter 8, "Using Counters and Timers," describes how to configure the counter/timer subsystem and use it as a clock, a trigger, to count events, and to measure a frequency.
- Chapter 9, "Developing Programs for VEE RunTime," provides some tips for developing programs that can be used in different hardware configurations with VEE RunTime.
- Chapter 10, "Advanced Functionality," details three methods of accessing DT VPI's advanced functionality: calling advanced functions in a Call Function object, using the unsecured versions of the data acquisition UserObjects, and integrating specialized thermocouple linearization UserFunctions into your own programs.
- Appendix A, "Example Programs," describes the examples provided with DT VPI.
- Appendix B, "Error Codes," describes the error codes that DT VPI can return.
- Appendix C, "File Formats," contains reference information about the file headers used in DT VPI.

- Appendix D, "Product Support," describes how to obtain technical support should you require it.
- An index completes this manual.

Conventions Used in this Manual

The following conventions are used in this manual:

When information applies to both VEE Pro and VEE OneLab, the term *VEE* is used. VEE provides the fundamental visual programming environment.

The term *DT VPI* refers to the component from Data Translation. DT VPI provides objects used to specifically control data acquisition hardware.

The term *click* (a menu selection, icon, button, or object,) means to move the mouse pointer to the appropriate location and quickly press and release the left mouse button.

The term *double-click* means to click the left mouse button twice in rapid succession.

The typographical conventions are listed in the table below.

Table 1: Convention s

Example of Convention	Description
DT VPI User's Manual	Italic is used for book titles and new terms.
Get Data Panel object	Helvetica font is used for names of objects.
DT DataAcq menu	**Bold** is used for menu names, text in dialog boxes, or any text you see on the screen.

Table 1: Conventions (cont.)

Example of Convention	Description
File ⇨ Open	The arrow ⇨ is used in a shorthand notation to show the location of VEE features in the menus. For example, **File** ⇨ **Open** means to select the **File** menu and then select **Open**.
Backspace	Represents a key on the PC keyboard that you should press.
Press Ctrl + A	Represents a combination of keys on the PC keyboard that you should press at the same time.

Related Documents

The following documents may help you to use DT VPI with VEE.

- DT VPI online help,
- VEE Multimedia tutorials,
- VEE online help,
- *VEE Pro User's Guide,*
- *VEE Pro Advanced Techniques,*
- *VEE OneLab User's Guide,* and
- *VEE OneLab Advanced Techniques.*

Where to Get Help

Should you run into problems installing or using DT VPI, our Technical Support Department is available to provide prompt, technical assistance. You will find their number in Appendix D. If you are outside the U.S. or Canada, call your local distributor, whose number is listed in your Data Translation product handbook.

Installing DT VPI

Step 1: Install Your Data Acquisition Board 3
Step 2: Install the Device Driver 4
Step 3: Install VEE 4
Step 4: Install DT VPI 5

This chapter tells how to install DT VPI. The installation is primarily automatic and requires only a few minutes if your computer is already set up and running.

There are four steps to get up and running. These four steps are described below and illustrated in Figure 1.

1. Install your data acquisition hardware, using the hardware user manual or getting started manual that comes with your board.

2. Install the device driver for your hardware, using the device driver documentation or getting started manual that comes with your board.

3. Install VEE Pro or VEE OneLab (version 6.0 or a later version), using the documentation that is provided with VEE.

4. Install DT VPI by following the steps outlined in the next section.

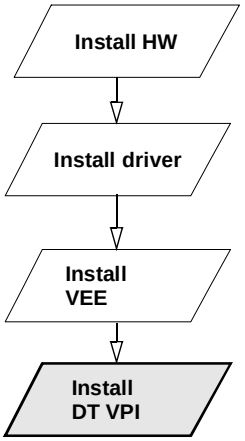

Figure 1: Installation Steps for VEE with DT VPI

Step 1: Install Your Data Acquisition Board

To use DT VPI, ensure that a data acquisition board from Data Translation is installed. Refer to the manual shipped with your board for information on installing the hardware.

Then, to perform the data acquisition examples in this manual, you must connect an external function generator or other signal source to your Data Translation data acquisition board. An oscilloscope, if available, can be used to display the output of the data acquisition board, but is not required. The following steps outline how to make these specific connections. For more information on connecting external input and output devices, consult your data acquisition hardware manual.

Connecting the Screw Terminal Panel

To connect the screw terminal panel to your installed data acquisition board, simply plug in the cable provided with the screw terminal panel as described in the following steps; your hardware user manual and/or screw terminal panel user manual has more information:

1. Insert one end of the cable to the J1 connector on the data acquisition board.

2. Insert the other end into the J1 connector on the screw terminal panel.

Connecting a Function Generator for Input

To connect a function generator or other signal generator to the screw terminal panel, perform these steps:

1. Attach the high side (positive, red or white) to the screw terminal panel's channel 0.

2. For differential inputs, attach the low side (negative, black) to channel 0 return.

   For single-ended inputs, attach the low side to any signal ground.

Connecting an Oscilloscope for Output

To connect an oscilloscope to the screw terminal panel, perform these steps:

1. Attach the high side (positive, red or white) to D/A 0 Out.

2. Attach the low side (negative, black) to D/A 0 Ground.

Step 2: Install the Device Driver

DT-Open Layers device drivers are provided on the Data Acquisition OMNI CD that is shipped with your board. Ensure that the WDM device driver for the operating system you are using is installed. Refer to the manuals shipped with your board for information on installing your device driver.

Step 3: Install VEE

Open the VEE package, which is included in the box along with DT VPI. Install VEE Pro or VEE OneLab, following the instructions in the VEE documentation.

Step 4: Install DT VPI

Once you have set up your computer, installed the data acquisition hardware and device driver, and installed VEE Pr or VEE OneLab, you are ready to install DT VPI, using the following instructions:

1. Start Microsoft Windows.

2. Insert Data Translation's DT VPI CD into an available CD-ROM drive.

3. Click **Start** located on the task bar, then click **Run**.
   The Run dialog box appears.

4. In the command-line text box, enter **D:\Setup.exe**, where D is the CD-ROM drive designation you are using for this installation, then click **OK**.
   The Install Shield Wizard dialog box appears.

5. Click **Next**.
   The license agreement dialog box appears.

6. Click **Yes**.

7. Select whether you want to use DT VPI with VEE Pro or with VEE OneLab, then click **Next**.
   The files are copied to your hard drive.

8. Click **Finish**.

Quick Start

The Basics 8
Working with Objects 14
Understanding Pins and Terminals. 22
Understanding Propagation 25

The Basics

In this section, you will learn how to start and quit VEE.You also will learn how to access the on-line help.

Starting VEE

To start VEE, follow these steps:

1. Click **Start** on the task bar.

2. Click **Programs**.

3. Click **VEE Pro 6.0 or VEE OneLab 6.0**.

When VEE is running, a button with the VEE logo appears on the task bar.

The VEE Pro window, shown in Figure 2, or the VEE OneLab window, shown in xx, appears. You can tell that DT VPI is installed because VEE's main menu has **DT DataAcq** present.

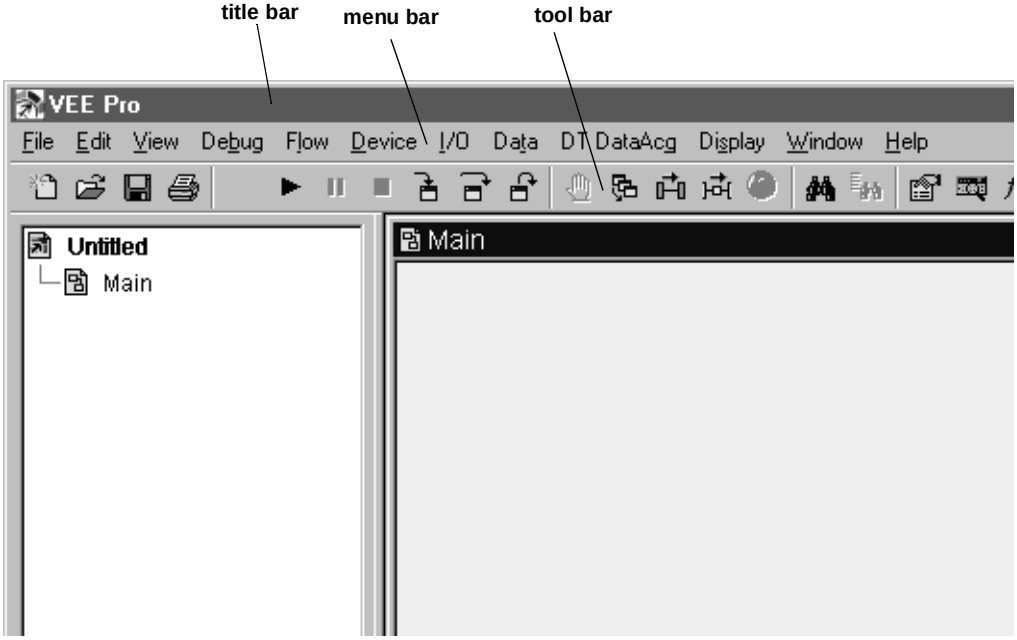

Figure 2: The VEE Pro Window with DT VPI Installed

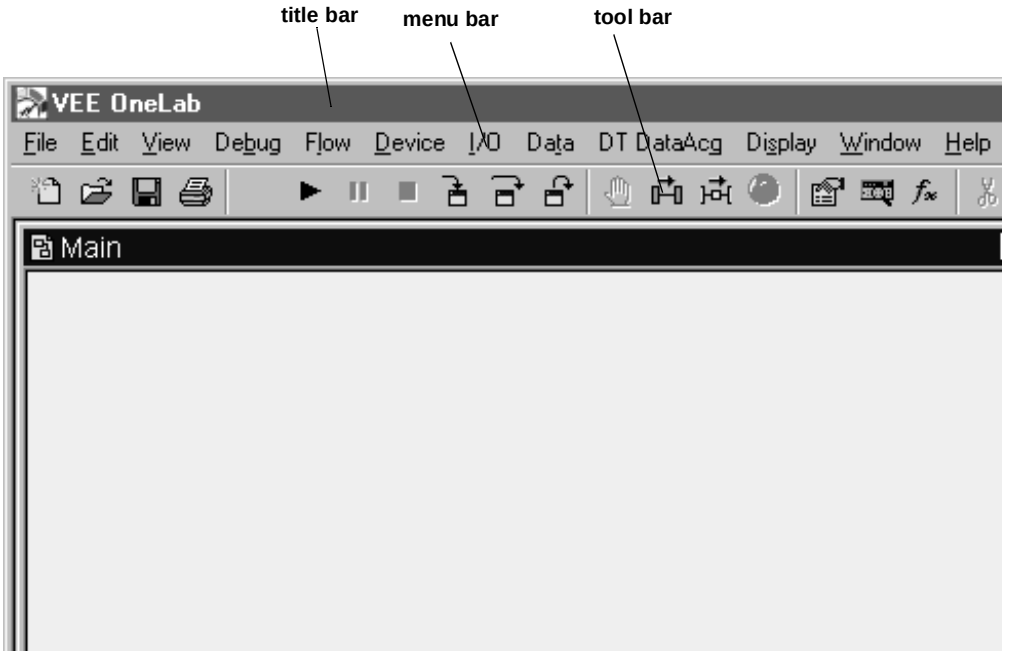

Figure 3: The VEE OneLab Window with DT VPI Installed

The four major areas of the VEE window are identified:

- The *title bar*. This is the standard Microsoft Windows title bar, which shows the name of the application, VEE. If you have opened an example program or saved your own program, then the program's file name also appears in the title bar. You can move the window by dragging the title bar.
- The *menu bar*, which enables you to select menu items and features.
- The *tool bar*, which provides handy buttons to **Run**, **Stop**, **Pause**, and **Step** VEE programs. The tool bar also has buttons for **New**, **Open**, **Save**, and **Print**, as well as others.

- The *work area*, which is the space where you create VEE programs.

You can move or resize the VEE window using standard Windows techniques.

Getting Help

VEE uses the standard Windows on-line help system, as does DT VPI. Both have their own help file.

- VEE help includes information on VEE's objects (essentially everything except data acquisition objects).
- DT VPI help includes information on all data acquisition objects and examples on using DT VPI.

Starting VEE On-line Help

To get on-line help for VEE's objects, menus, tasks, and examples, click **Help** from the main menu, then click **Contents and Index**.

Starting DT VPI Online Help

To access on-line help for the data acquisition objects and examples, you can do one of the following:

- Click **Start** on the task bar, click **Programs**, select **Data Translation\DT VPI 6.0**, then click **DT VPI Help**, or
- Select a DT VPI object in the work area by clicking it. Click the right mouse button and then select **Help** from the object menu.

If you select **DT VPI Help**, the window shown in Figure 4 appears.

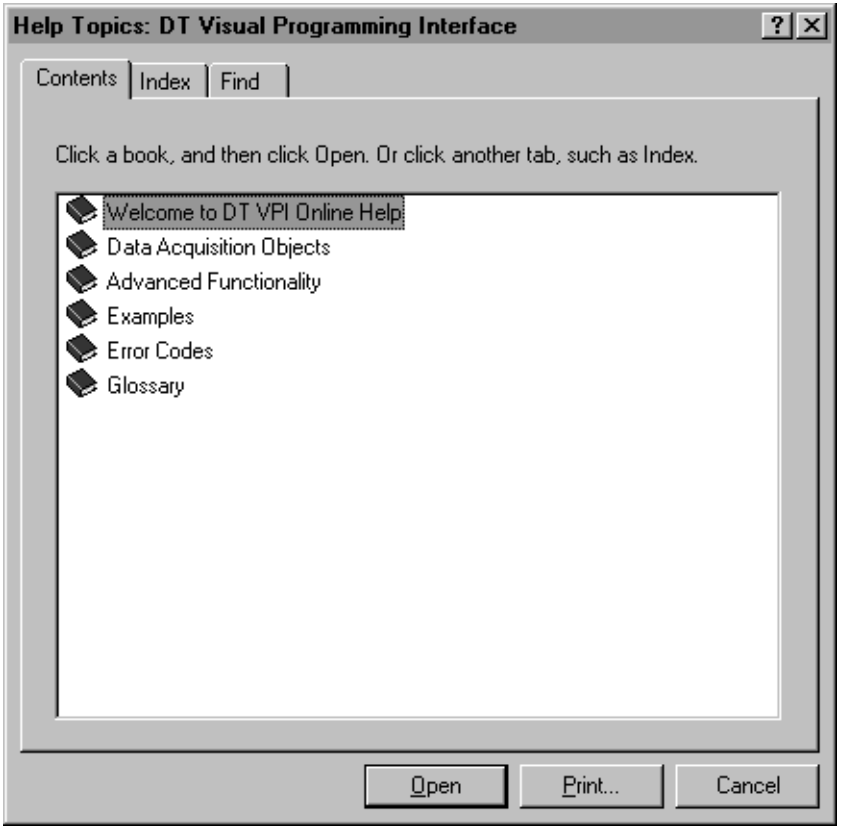

Figure 4: Contents of DT VPI Online Help

Double-click a book icon to see the help topics available. The major topics listed in the Contents screen are as follows:

- **Welcome to DT VPI On-line Help** provides an overview of DT VPI and its relationship to VEE.
- **Data Acquisition Objects** describes each of the objects and menus provided with DT VPI. They are listed alphabetically and by subsystem.
- **Advanced Functionality** describes three ways to tinker with the inner workings of DT VPI.
- **Examples** describes the example programs included with DT VPI.
- **Error Codes** lists all error codes output from an error terminal and the corresponding message.
- **Glossary** provides a brief definition of terms referenced in this manual.

To quit help and close the help window, click the X in the upper right corner of the help window.

Quitting VEE

You can quit VEE by double-clicking the VEE icon in the upper left corner of the window or by selecting **File ⇨ Exit**. However, if it is running, stop the program first by clicking the **Stop** button ■ on the tool bar.

If you have not saved your most recent changes to a file, a dialog box asks you if you want to save your changes. Select **No** to discard your changes, **Yes** to save them to a file, or **Cancel** to continue working in VEE.

Working with Objects

In this section you will learn the basics of manipulating objects. Clear your work area by selecting **File ⇨ New**.

Adding an Object to the Work Area

In VEE, each object is located in one of the menus. To add a new object to the work area, select a menu, select one of the menu items, and then click on the work area to place the object. Try it now:

1. Select the **Device** menu.
   A drop-down menu appears.

2. Move the mouse pointer down to the **Virtual Source** item.
   A submenu appears.

3. From the submenu, select **Function Generator**.
   The menu closes and an outline of the object appears.

4. Move the mouse cursor anywhere in the work area, and click once to place the Function Generator object, shown in Figure 5, in the work area.

Figure 5: The Function Generator Object

You can move, cut, paste, clone, iconize, change the title of, and resize objects with just a few clicks of the mouse. These techniques apply to all objects in VEE, not just those in DT VPI.

Moving an Object

To move the Function Generator object:

1. Click anywhere on the object (except in text edit boxes) and hold down the mouse button.
   An outline of the object appears while you drag the mouse.

2. Move the outline to the middle of the work area while continuing to hold the mouse button down.

3. Release the mouse button.
   The object jumps to the new location.

Selecting the Object Menu

Each object has an *object menu*, which enables you to change attributes

such as the size, position, and title of the object. Also, you can get online help for the specific object from the object menu. To select the object menu, perform these steps:

1. Position the mouse over the object.

2. Click the right mouse button.
   The object menu appears, as shown in Figure 6.

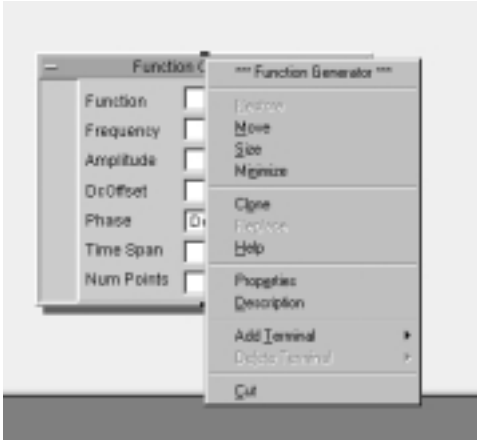

Figure 6: The Function Generator's Object Menu

Now you can click one of the object menu choices to perform the action you desire. To make the object menu disappear, click an empty area *outside* the menu.

Copying an Object

To copy the Function Generator object and save it in a buffer for pasting late, perform these steps:

1. Select the object.

2. Click the **Copy** button .
   The object goes to a buffer for pasting later. It is not deleted from the work area.

Each time you use Copy or Cut, whatever is in the buffer is overwritten. If it is running, stop the program before cutting or copying.

Note: The buffer used to copy and paste VEE objects is not the Windows clipboard.

Cutting an Object

To delete the Function Generator object and save it in a buffer for pasting later, perform these steps:

1. Select the object.

2. Click the **Cut** button .
   The object is deleted from the work area and goes to a buffer for pasting later.

Other ways to cut an object are as follows:

- Select the object, click the right mouse button to get the object menu, and then click **Cut**.
- Double-click the control bar of an object in open view.
- Press Ctrl + D when the mouse cursor is over the object.

Each time you use Copy or Cut, whatever is in the buffer is overwritten. If it is running, stop the program before cutting or copying objects.

Pasting an Object

Once an object is in the buffer, you can paste the object as many times as you want. To paste the object in the buffer onto the work area, perform these steps:

1. Click the **Paste** button .
   An outline of the object appears.

2. Move the outline to the desired location in the work area.

3. Click to place the object.

Changing Object Views

Every VEE object has two views—an *open view* and an *icon*—from which you can choose. When first accessed, an object, by default, appears in its open view. In the open view, all object parameters are visible and you can edit them. In icon view, the object is minimized, much like you can minimize a window in Windows. The *minimize button* is located at the right end of the object's title bar.

Change a Function Generator object from its open view to its icon by clicking once on the minimize button. You can use the icon to save space in your work area. To return to the open view, double-click the icon. To change parameters, the object must be in open view. The two views of the Function Generator object are shown in Figure 7.

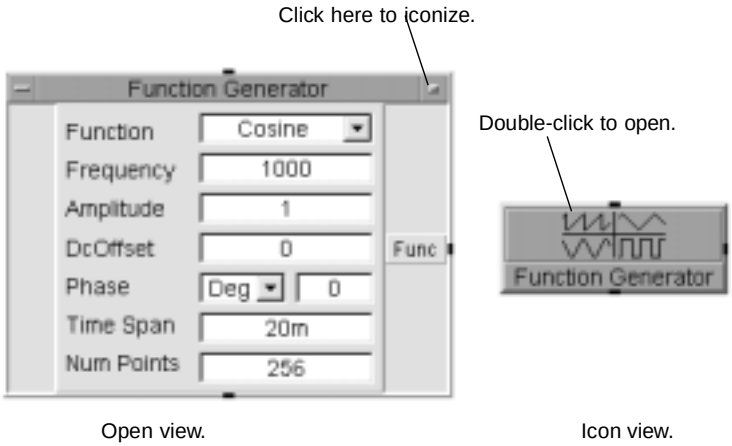

Figure 7: Open View and Icon View of an Object

Changing the Size of an Object

You can change the size of an object in open view or icon. To change the size of an object, perform these steps:

1. Position the mouse pointer over the lower right-hand corner of the object until you see a left angle.

2. Click and drag to the desired size.

Or,

1. Select **Size** from the object menu.
   The mouse pointer becomes a left angle.

2. Move this bracket (as shown in Figure 8) to define the size that you want, then click.

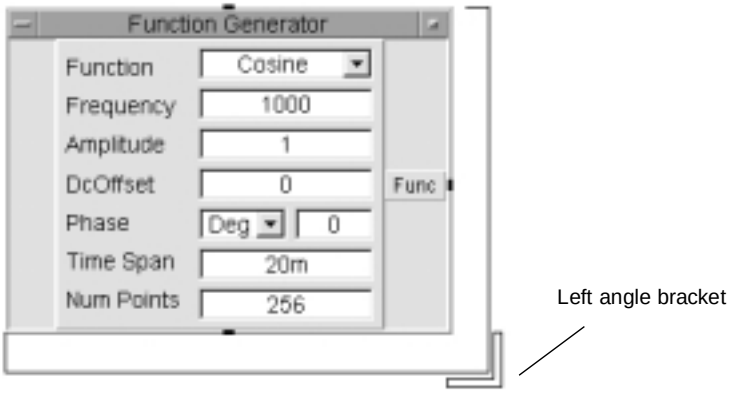

Figure 8: Resizing an Object

(Note that this procedure is a little different than that for resizing a window.)

Try resizing the object a few times on your own. You can enlarge the object or reduce it as much as you want.

Changing the Title of an Object

To change the title of an object, perform these steps:

1. Select **Edit Properties** from the object menu.
   The Edit Properties dialog box shown in Figure 9*, appears.*

2. Type the new title and click **OK**.

To enter a new title, start typing over the highlighted text. When you click **OK**, the new title appears in the title bar. If you minimize the object, the new title labels the icon.

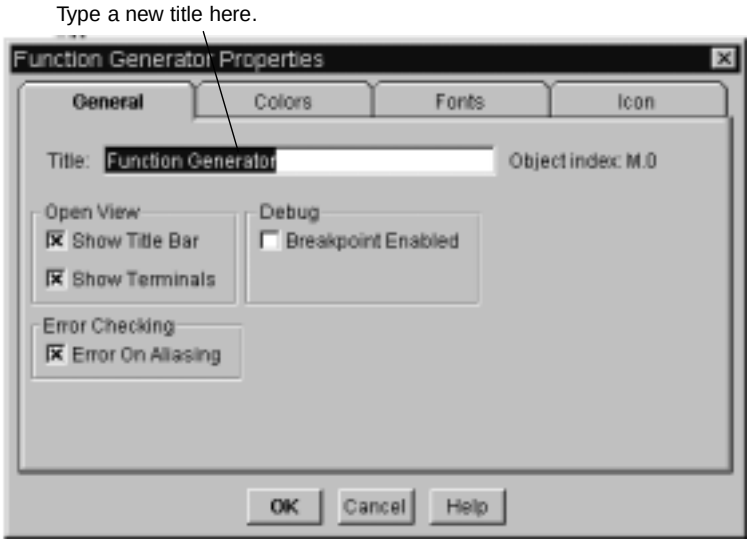

Figure 9: The Edit Properties Dialog Box

Note: You can save time by using standard Windows editing techniques. For example, the left- and right-arrow keys move the cursor, and the Delete and Backspace delete letters, as you would expect.

Understanding Pins and Terminals

A single object can do little by itself. However, by connecting objects together with lines, you can construct a VEE program. The points of connection are the input and output *pins*. Most objects have one or more of the following kinds of pins, as shown in Figure 10:

- Pins on the left-hand side of an object are *data input pins*.
- Pins on the right-hand side of an object are *data output pins*.
- The pin on the top of an object is the *sequence input pin*.
- The pin on the bottom of an object is the *sequence output pin*.

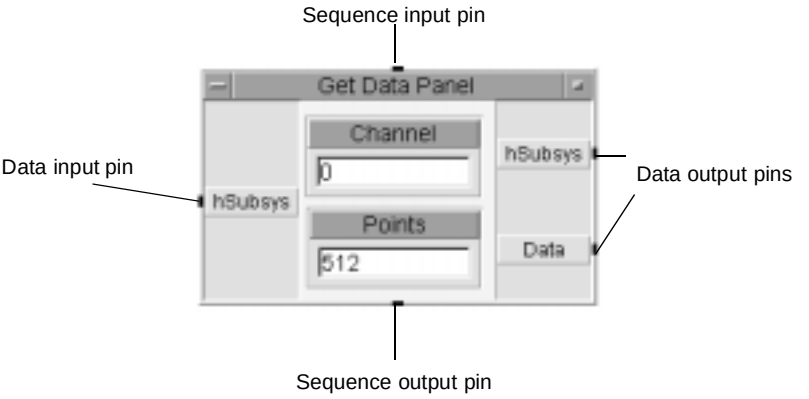

Figure 10: The Various Pins on an Object

Some objects have all four kinds of pins, while others have only one or two kinds of pins. In an object's open view, the data input and output pins can be represented as input and output *terminals*, which display their *terminal labels*. The terminal labels are visible only in the open view, and only if the Show Terminals option is active.

Connecting Objects

Data flows from one object to another through connections, from left to right. A data flow connection is made between the data output pin of the first object and the data input pin on the second object.

To connect two objects, first place the objects on the work area:

1. Click the left mouse button just outside of the data output pin of the first object.
   Do not click on the pin itself or inside the object. When it is in the right position, the cursor turns into a small square □.

2. Position the mouse pointer near the data input pin on the other object. *When it is near an available data input pin, a small square appears around the input pin.*

3. Click.
   A line is automatically routed between the two pins; the connection is complete.

Adding Data Input Terminals to an Object

You can add an input terminal to an unlocked object if you need one by performing one of these steps:

- Open the object menu (right mouse button over the object) and select **Add Terminal** ⇨ **Data Input**. The terminals are automatically named A, B, C,..., but you can rename them.
- Just place the mouse pointer over the terminal input area and press Ctrl+A.

Deleting Data Input or Output Terminals from an Object

To delete data input or output terminals from an object, perform these steps:

1. Open the object menu, and select **Delete Terminal ⇨ Input...**.
2. Choose the input to delete.
3. Click **OK**.

Or, you can just place the mouse pointer over the terminal and press Ctrl + D.

Renaming a Terminal

To rename a terminal, perform these steps:

1. Double-click the terminal.
   *The **Name** input field is highlighted.*
2. Type the new name, and click **OK**.

Understanding Propagation

In conventional programming languages such as C, BASIC, or Pascal, the order in which program statements are executed is determined by a set of sequence and selection rules. Generally, statements are executed in the order they appear in the program, except where statements cause execution to branch to another statement or section of code.

In VEE, the general flow of execution through a program is called *propagation*. Propagation through a program is not determined by the geographic locations of the objects in the program, but rather by the way the objects are connected. Propagation is primarily determined by *data flow*, which is determined by how the data input and output pins of the objects are connected.

To understand how the data flows in a VEE program, perform these steps:

1. Open an example program. To do this, select **Help ⇨ Open Example**, double-click the **DataAcq** folder, then double-click the program ADSIMPLE.VEE.
   If you do not have a signal source connected, then select a program in a different folder.

2. Click the **Show Data Flow** button on the tool bar ( ).
   If this icon is not present, then you are in panel view. Switch to detail view by clicking .

3. Run the program (click), watching for the small square traveling along the connections between the objects. The square represents a packet of data.

To see the order in which the objects execute, perform these steps:

1. Click the **Show Execution Flow** button on the tool bar ( ).

2. Run the program (click).
   When an object executes, a border blinks around the object for a short time.

In the ADSIMPLE.VEE program, the data and execution flows like this:

1. First, the A/D Config object is executed.
   *If you are using hardware, the A/D Config object configures the hardware to sample data at a specified rate on a specified channel. If you are using Simple Driver, the configuration is simulated. When it is done executing, the A/D Config object outputs a subsystem handle on its data output pin. The subsystem handle (**hSubsys**) identifies the current configuration of the A/D subsystem to any data acquisition object that needs to interact with it.*

2. A data packet carrying the subsystem handle travels along the connection from the A/D Config object to the Get Data Panel object.

3. The Get Data Panel object is executed when it receives the subsystem handle on its data input pin. It automatically starts the board acquiring. *When enough data is in memory, the Get Data Panel object retrieves some data. The exact number of data points it retrieves is specified in the **Points** edit box. (Open the Get Data Panel object to see the **Points** box.)*

4. When it has retrieved as many samples as is specified in the Points edit box (in this case, 512), the Get Data Panel object outputs an array containing the data on its data output pin.
   *You can see the packet traveling along the connection between the Get Data Panel object and the Waveform (Time) object. To see this data, click the **Line Probe** button on the tool bar and click your mouse on the connection between the two objects.*

5. When it receives data on its input pin, the display object is executed, displaying the data.

Here are the rules of data flow through objects:

- *Data flows from left to right through an object.* This means that on all objects with data pins, the left data pins are inputs and the right data pins are outputs.
- *All of an object's data input pins must be connected.* Otherwise, an error occurs when the program is run.
- *An object will not be executed until all of its data input pins have received data.*
- *An object finishes executing only after all appropriate data output pins have been activated.*

In VEE you can change the order of execution by using sequence input and output pins. If possible, let data flow control the execution of your program.

In general, the rules of data flow through a program are as follows:

- *Start objects operate first.* The Start object is used to operate an individual thread.
- *Objects with no input pins operate next.*
- *Objects with data input pins operate when all connected input pins are satisfied.* Recall that connecting sequence input pins is optional.

Your First Program

Lesson 1: Create A Simple Program 31
Lesson 2: Modify Your Simple Program 38
Lesson 3: Acquire Analog Data 43
Lesson 4: Acquire with Get Single Value 50
Lesson 5: Output Analog Data 60
Lesson 6: Digital Input and Output 66
Lesson 7: Change the Configuration Programmatically 70
What's Next? 78

Now that you have learned to work with objects and the VEE user interface, we will look at how to use objects to create VEE programs.

This chapter contains seven lessons, each with several steps. By completing these tutorial lessons, you will learn how to

- Create, save, and run a simple program;
- Acquire analog data;
- Determine whether your hardware supports gap-free continuous data transfer;
- Output analog data;
- Acquire digital input data; and
- Change a subsystem's configuration using objects in the program.

Lesson 1: Create A Simple Program

6 minutes

In Lesson 1 you will create a simple VEE program that displays a generated waveform. You will also print the VEE screen and save your program to a file.

Restart VEE if you previously quit it. If VEE is already running, clear the work area by selecting **File ⇨ New**.

Step 1: Add the Objects

Fundamentally, a VEE program consists of VEE objects connected to form an executable *block diagram*. We will create a very simple program that displays a waveform:

1. Select **Device ⇨ Virtual Source ⇨ Function Generator**.
2. Place the Function Generator object on the left part of the work area.
3. Select **Display ⇨ Waveform (Time)**.
4. Place the Waveform (Time) object on the right of the work area, as shown in Figure 11.

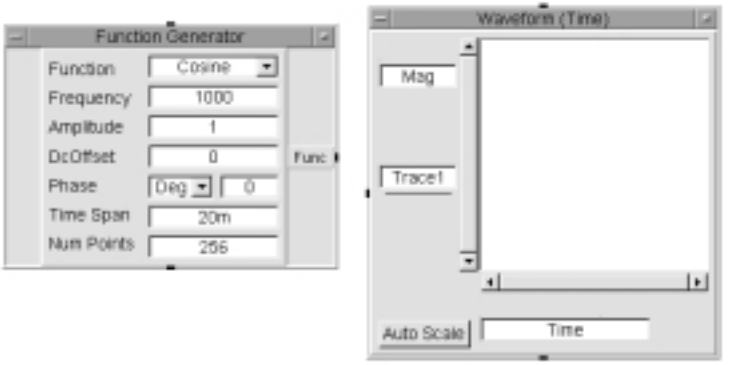

Figure 11: The Objects in a Simple Program

Note: The tic marks and axes in the Waveform (Time) object have been removed in Figure 11 and in the remaining figures.

Step 2: Connect the Objects

Data flows from one object to another through *connections*. For this lesson, make a correction to the Waveform (Time) object so that data flows from the Function Generator object to the Waveform (Time) object:

1. Click the left mouse button just outside of the data output pin on the Function Generator object.
   When your cursor is near the pin, a small square appears around the pin.
2. Click the mouse pointer again near the data input pin on the Waveform (Time) object.

A line is automatically routed between the two pins; your program is complete.

Step 3: Run Your Program

Once you have placed the objects you need on the work area and connected them, you are ready to run the program:

1. Click the **Run** button ▶ in the tool bar.
The program displays a 1000 Hz cosine wave in the Waveform (Time) object, as shown in Figure 12.

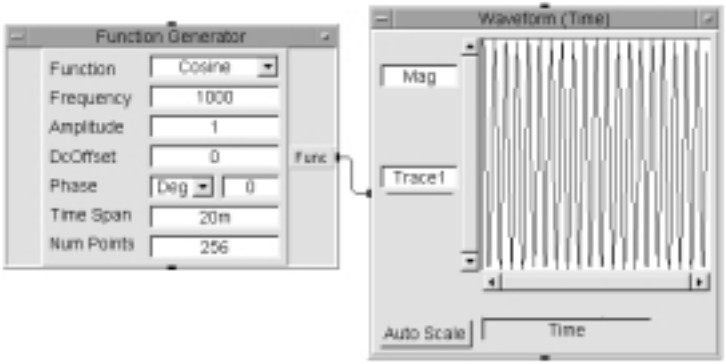

Figure 12: The Output from the Simple Program

Note: From now on, when we say to *run* your program, just click the **Run** button in the tool bar ▶ (or press Ctrl + G).

Step 4: Change Object Parameters

You can change the parameters of an object in its open view, as follows:

1. In the Function Generator object, click **Cosine** (or the arrow) in the Function field.
   A drop-down list of selections, shown in Figure 13*, appears.*

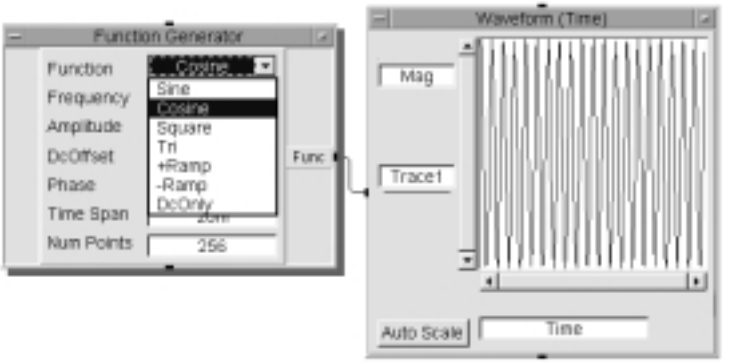

Figure 13: Changing Cosine to Sine

2. Click **Sine** to select the sine function.

3. Edit the Frequency field to read 100 instead of 1000, as shown in Figure 14.

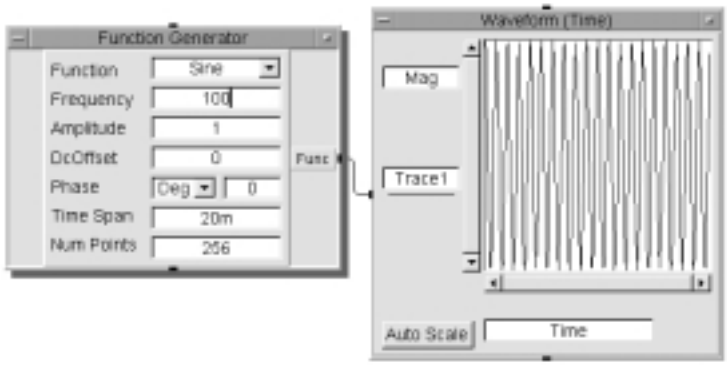

Figure 14: Changing the Frequency from 1000 Hz to 100 Hz

4. Run the program. The displayed waveform is now a 100 Hz sine wave, as shown in Figure 15.

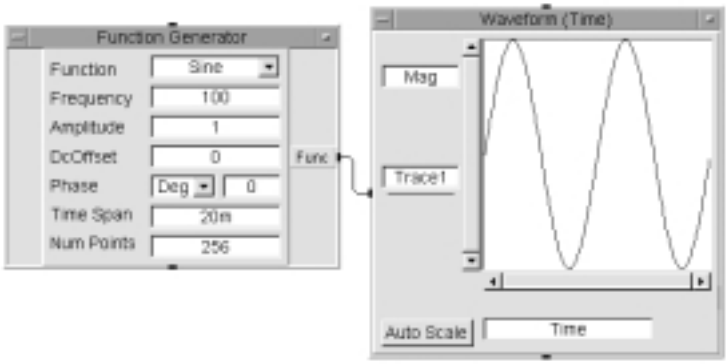

Figure 15: The Output from the Program

Try changing a few object parameters on your own:

- Click **Deg** (or the arrow) in the Function Generator object and change the phase units to **Rad**. Now, click the **Phase** value field and enter the value **PI**. Click the **Run** button and note the phase shift in the displayed waveform. Now change the **Phase** value back to **0** and the units back to **Deg**.
- The y-axis limits of the Waveform (Time) object are preset to –1 through 1. Click each value field and change the limits to –2 through 2. you will see the waveform displayed within the new limits. Now, click **Auto Scale;** the waveform is automatically scaled from –1 to 1.

Step 5: Print the Screen

To print the screen, click the **Print** button on the tool bar. VEE prints the VEE window on the printer indicated in the Print setup dialog box.

Step 6: Save Your Program

You can save your program (whatever is in the work area, whether a complete program or not) at any time, by performing the following steps:

1. Click the **Save** button .
   By default, VEE saves your files to the MY DOCUMENTS \ VEE PROGRAMS directory.

The Save File dialog box, shown in Figure 16, appears.

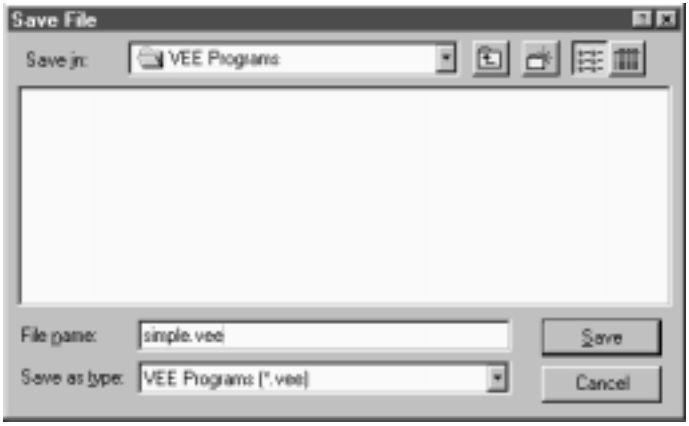

Figure 16: Saving a VEE Program to a File

2. Enter a name (for example, SIMPLE), and click **Save**.
VEE automatically adds the extension.VEE to the file name.

Note: To re-save your program to the same file name, select **File** ⇨ **Save** (or click 💾 or press Ctrl + S) at any time. It is a good idea to save your file frequently while you are developing a program.

Lesson 2: Modify Your Simple Program

5 minutes

In Lesson 2, you will modify the program you created in Lesson 1. You will add noise to the waveform, and then add an adjustable amplitude input to the noise.

Restart VEE if necessary, and open the program from Lesson 1, SIMPLE.VEE (by selecting **File ⇨ Open**).

Step 1: Make Some Noise

Add some noise to the waveform:

1. Click the **Delete Line** button on the tool bar .

2. Click the connection between the Function Generator object and the Waveform (Time) object.
   The connection disappears.

3. Minimize the Function Generator object to an icon. (To do this, click the minimize button in the upper right corner of the object.)

4. Select **Device ⇨ Virtual Source ⇨ Noise Generator**, and place the Noise Generator object below the Function Generator object.

5. Select **Device ⇨ Formula, and place it between the** Function Generator **and** Waveform(Time) **object.**

6. Click your right mouse button and select AddTermina ⇨ ata Input.

7. Delete the current formula and replace it with a+b.

8. Connect the objects as shown in Figure 17.

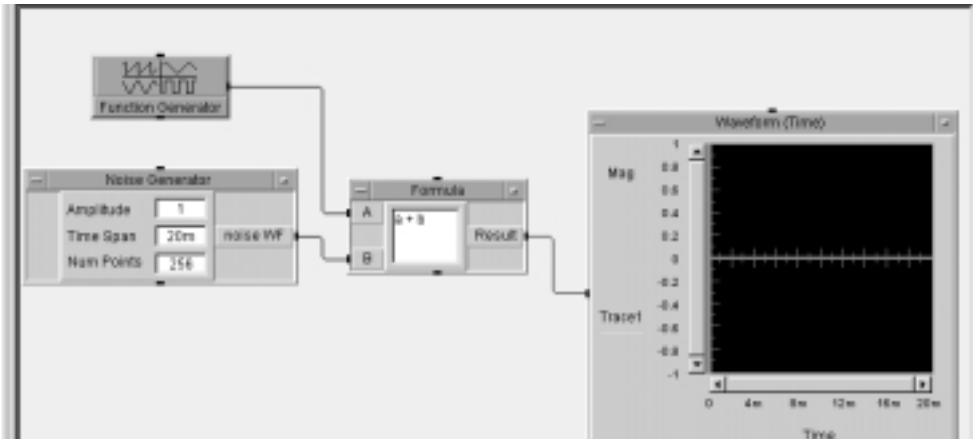

Figure 17: Adding a Noise Generator to the Program

9. Run the program.

10. Click **Auto Scale** on the Waveform (Time) object.
The output is shown in Figure 18.

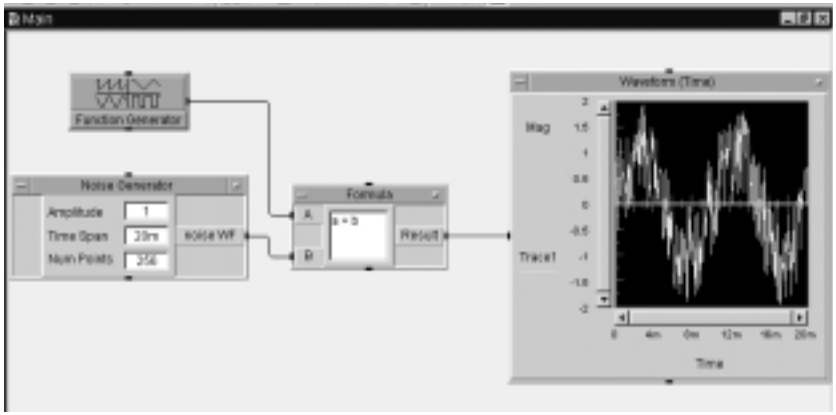

Figure 18: A Sine Wave with Added Noise

Step 2: Add an Amplitude Input

Now, add an amplitude input for the Noise Generator object:

1. Click the Noise Generator object to select it.
   A shadow appears behind the object when it is selected.

2. Position the mouse pointer in the *terminal area* at the left side of the Noise Generator object.
   The terminal area is the sunken area to the left of the words "Amplitude," "Time Span," and "Num Points."

3. Press Ctrl + A.

4. A dialog box asks you to select an input to add, as shown in Figure 19:

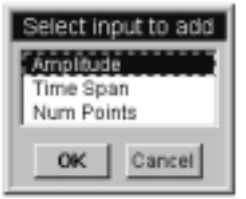

Figure 19: Adding an Amplitude Input to the Noise Generator

5. Select **Amplitude** by clicking **OK**.
   An amplitude input terminal appears.

Note: If you need more space to the left of the Noise Generator object, drag the work area to the right.

---

6. Select **Data** ⇨ **Continuous** ⇨ **Real Knob** and place the knob to the left of the Function Generator object.

7. Connect the data output pin of the Real Knob object to the data input terminal of the Noise Generator object, as shown in Figure 20.

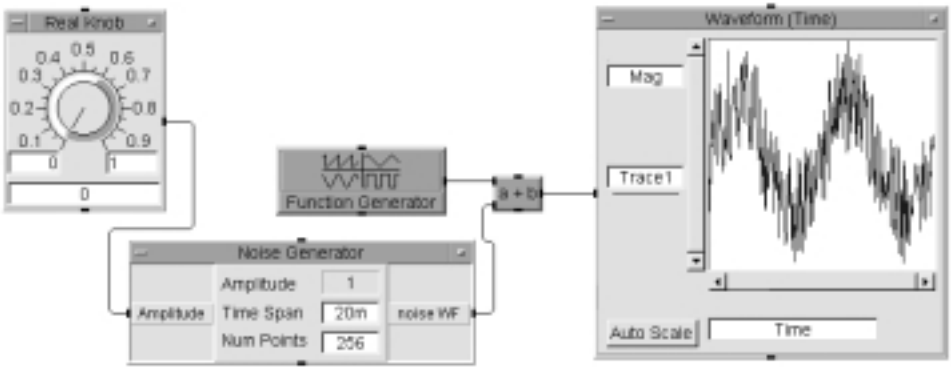

Figure 20: Adding a Knob

Step 3: Run!

Run the program:

1. Click the **Run** button ▶.
   By default, the noise amplitude is 0.

2. Adjust the amount of noise by "turning" the knob with your mouse.

3. Run the program again. With the noise amplitude set at.65, the output looks something like that shown in Figure 21.

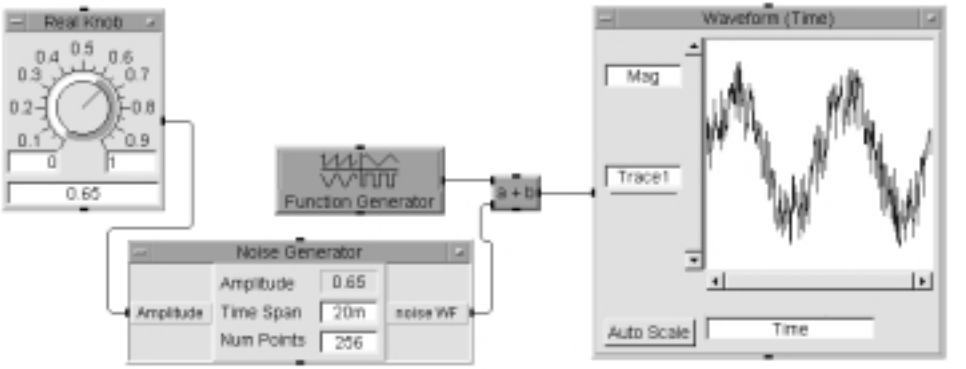

Figure 21: Output when User Sets the Noise Level to.65

4. Save your program.

Data flow determines the order of execution. The Noise Generator object is not executed until the Real Knob object executes. The a + b object is not executed until both the Function Generator and the Noise Generator objects have executed, but it does not matter which one executes first. Finally, the Waveform (Time) object executes only after the a + b object has executed.

Lesson 3: Acquire Analog Data

8 minutes

You can easily incorporate data acquisition into your VEE programs using the objects found in the **DT DataAcq** menu.

In Lesson 3 you will create a simple program to acquire analog data from an external source, such as a function generator, and display it. Then, you will add an Until Break object to acquire continuously.

Restart VEE if you quit previously. If VEE is already running, clear the work area by clicking the **New** button on the tool bar .

Note: If your board does not support continuous data acquisition with gap-free data, you cannot use the Get Data Panel object when acquiring data. Lesson 4 covers how to use the Get Single Value object instead.

To determine if your board supports continuous data acquisition with gap-free data, locate the device driver documentation that accompanies your data acquisition hardware. InTable 1, “Supported Options,” check the row labelled “Continuous Operation Support.” If there is “Yes” in the column under the A / D subsystem, then your board can continuously acquire data using the Get Data Panel object. If not, then you must use the Get Single Value object instead.

Step 1: Gather the Equipment

Before beginning to write data acquisition programs in VEE, make sure you have the equipment needed to perform analog input. For this lesson you need

- An installed data acquisition board from DataTranslation and its Windows device driver (refer to your hardware and device driver documentation).

- Insure that you have the latest WDM drivers from Data Translation. You can find them on our web site. (www.datatranslation.com)
- A screw terminal panel connected to the data acquisition board in your system (refer to your screw terminal panel documentation).
- A function generator or other signal source connected to A/D channel 0 on the screw terminal panel. Make sure the signal source is turned ON.

Step 2: Add the Objects

To perform analog input, add one object that configures the hardware, one object that gathers the data, and one object that displays the data:

1. Select **DT DataAcq ⇨ A/D Config** and place the A/D Config object in the left of the work area.
2. Select **DT DataAcq ⇨ Data ⇨ Get Data Panel** and place this object to the right of the A/D Config object.
3. Select **Display ⇨ Waveform (Time)** and place it to the right of the Get Data Panel object.

4. Connect the objects as shown in Figure 22.

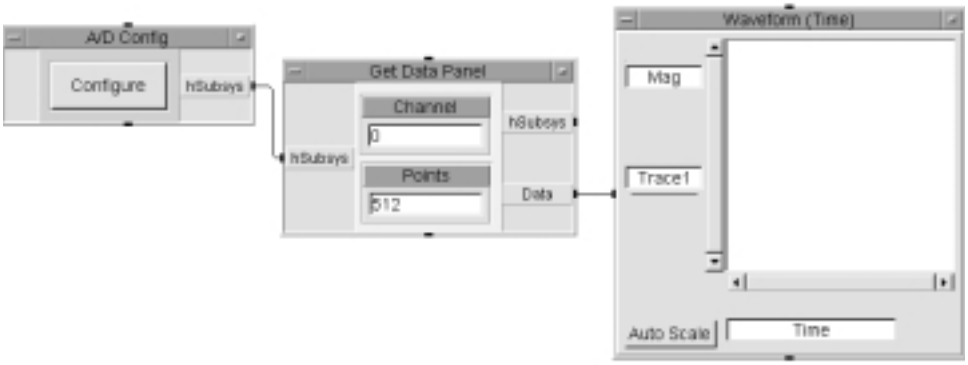

Figure 22: A Simple Analog Input Program

Step 3: Configure the A/D Subsystem

Before you can run the program, you must configure the A/D subsystem on the data acquisition board. A typical data acquisition board has more than one subsystem on it; usually, a data acquisition board has an analog-to-digital (A/D) and digital-to-analog (D/A) subsystem, as well as possibly a Counter/ Timer (C/T) subsystem, digital input (DIN) subsystem, and digital output (DOUT) subsystem.

In this lesson, we will configure the A/D subsystem, which performs the analog data acquisition from the external signal source, converting it to digital form for use by VEE:

1. Click the **Configure** button on the A/D Config object. The A/D Configuration dialog box appears, as shown in Figure 23.

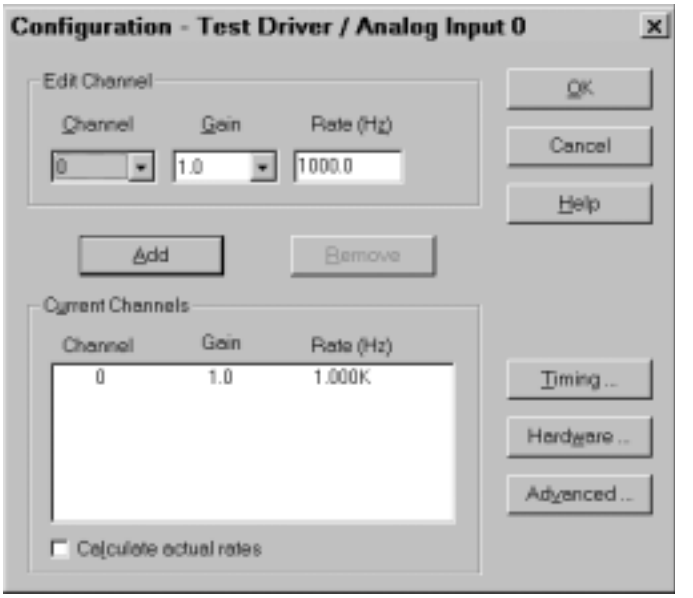

Figure 23: Configuring the A/D Subsystem

Note: If you have more than one data acquisition board installed or more than one instance of that board, a dialog box asks you to indicate which board to use. You can access only one board at a time.

The A/D Configuration dialog box enables you to specify which channels you wish to use for data acquisition, and the sampling rate and gain for each.

The default configuration in the dialog box specifies that channel 0 is sampled at a rate of 1000 Hz (samples per second) and at a gain of 1. These settings are sufficient for our example.

2. Click **OK** to close the dialog box and to configure the A / D subsystem with these settings.

Your program is ready to run. Make one last check to be sure your equipment is correctly connected and turned on.

Step 4: Run!

To run your program, perform these steps:

1. Set the external function generator to output a waveform (a sine waveform is fine) at a frequency of 100 Hz.

   We configured the A / D subsystem to sample data at a rate of 1000 Hz. Therefore, according to the Nyquist theorem, the highest frequency component of the input signal must be less than half of 1000 Hz to avoid aliasing. Since the input signal is well under 500 Hz, we do not have a risk of aliasing.

2. Run the program by clicking the **Run** button .

3. Click **Auto Scale** on the Waveform (Time) object to see the output better. The output is shown in Figure 24.

Figure 24: 512 Samples of Acquired Data

Step 5: Acquire Continuously

You can repeat the analog input continuously until you press the **Stop** button. To do this, we will add an Until Break object.

1. Select **Flow** ⇨ **Repeat** ⇨ **Until Break** and place the Until Break object above the Get Data Panel object.

2. Connect the A/D Config object's sequence output pin to the Until Break object's sequence input pin.
   This connection ensures that the A/D subsystem is configured before the acquisition loop begins.

3. Connect the Until Break object's data output pin to the Get Data Panel object's sequence input pin, as shown in Figure 25.

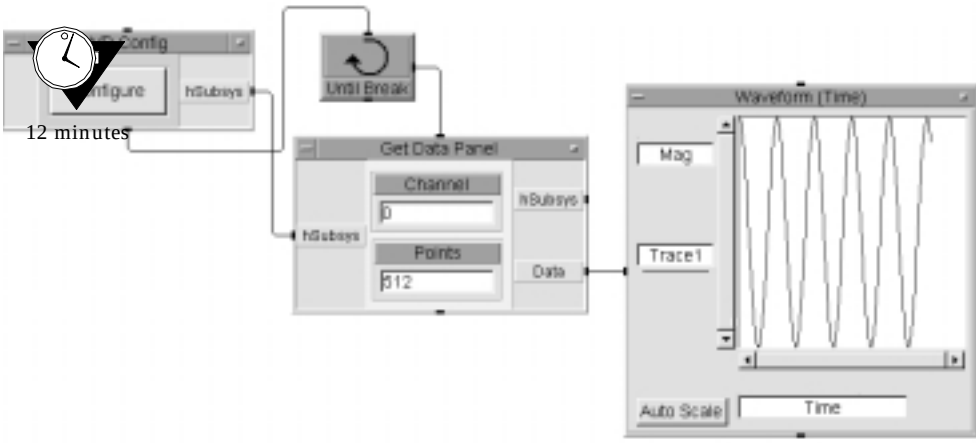

Figure 25: Acquiring Data Continuously

4. Run the program.
The Waveform (Time) object displays 512 points of the acquired signal at a time.

5. When you are ready to stop the program, click the **Stop** button . *Acquisition actually continues for a moment or two, until the memory buffers on the hardware are filled.*

Step 6: Save Your Program

It is a good idea to save your program often. You will use this program again in Lesson 7. To save your program, follow these steps:

1. Click the **Save** button .

2. In the Save File dialog box that appears, enter the name AD.VEE.

3. Click **Save** to close the dialog box.

Lesson 4: Acquire with Get Single Value

Acquiring a single value of data is typically used when you are not concerned with a pacer clock for acquisition, such as when you are testing the output of a battery, the temperature of an oven, or some other constant or slowly changing signal.

In Lesson 4 you will use the Get Single Value object to repeatedly acquire a single value until you click the Stop button. If your board does not support continuous data acquisition with gap-free data, you cannot use the Get Data Panel object when acquiring data, so you must use the Get Single Value object instead. In this case, by repeatedly acquiring a single value, you can simulate clocked data acquisition.

Restart VEE, if necessary. If VEE is already running, clear the work area by clicking the **New** button on the tool bar .

Step 1: Gather the Equipment

Before beginning this lesson, make sure you have the equipment needed to perform analog input. For this lesson you need

- An installed data acquisition board from DataTranslation and its Windows device driver (refer to your hardware and device driver documentation).
- A screw terminal panel connected to the data acquisition board in your system (refer to your screw terminal panel documentation).
- A function generator or other signal source connected to A/D channel 0 on the screw terminal panel. Make sure the signal source is turned ON.

Step 2: Add the Objects

To perform analog input, add one object that configures the hardware, one object that gathers the data, and one object that displays the data:

1. Select **DT DataAcq** ⇨ **A/D Config** and place the A/D Config object in the left of the work area.

2. Select **DT DataAcq** ⇨ **Data** ⇨ **Get Single Value** and place this object to the right of the A/D Config object.
   The Get Single Value object acquires one sample from the A/D subsystem immediately upon executing. It does not wait for a pacer clock, so a sampling rate is not needed.

3. Select **Data** ⇨ **Constant** ⇨ **Int 32** and place the Int 32 object to the left of the Get Single Value object.
   This object provides the channel number.

4. Select **Flow** ⇨ **Repeat** ⇨ **Until Break** and place the Until Break object above the Get Single Value object.

5. Select **Display** ⇨ **Strip Chart** and place this object to the right of the Get Single Value object.

6. Connect the objects as shown in Figure 26.

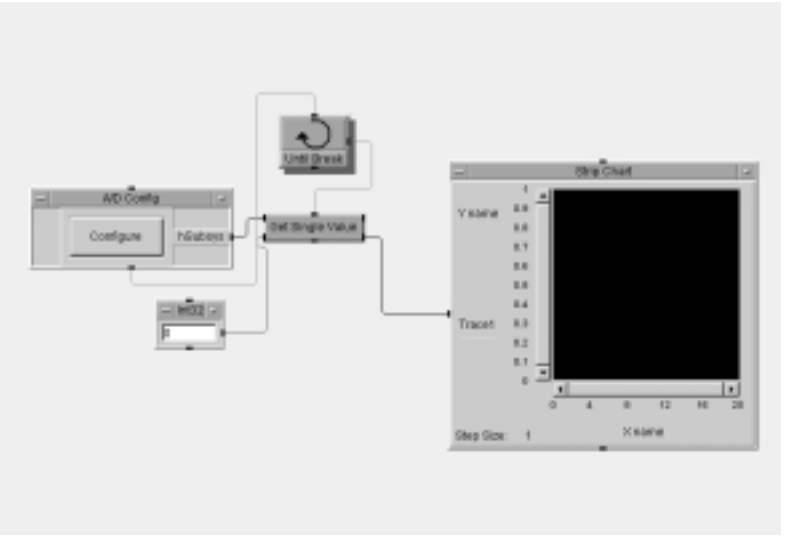

Figure 26: An Analog Input Program Using Get Single Value

Step 3: Configure the A/D Subsystem

As in the last lesson, we will be using the A/D subsystem, which performs the analog data acquisition from the external signal source, converting it to digital form for use by VEE.

Before you can run the program, you need to configure the A/D subsystem on the data acquisition board; however, in this lesson, the A/D Config object is used only to configure a subsystem handle (the channel list, sampling rate, and gain have no meaning when using Get Single Value):

1. Click the **Configure** button on the A/D Config object.
   The A/D Configuration dialog box appears.

Note: If you have more than one data acquisition board installed or more than one instance of that board, a dialog box asks you to indicate which board to use. You can access only one board at a time.

2. Since the sampling rate, gain, and channel list are meaningless when using Get Single Value, just click **OK** to close the dialog box and to configure the A/D subsystem.

Note: The Get Single Value object uses the hsubsys from the A/D Config object, but does not use the channel list. The Integer Constant object passes the channel number and the appropriate gain is automatically used to acquire the signal.

Your program is ready to run. Make one last check to be sure your equipment is connected correctly and turned on.

Step 4: Run!

To run your program, perform these steps:

1. Set the external function generator to output a waveform (a square wave will work nicely) at a frequency of 100 Hz.

2. Run the program.

3. The output is displayed in the Strip Chart object, as shown in Figure 27.

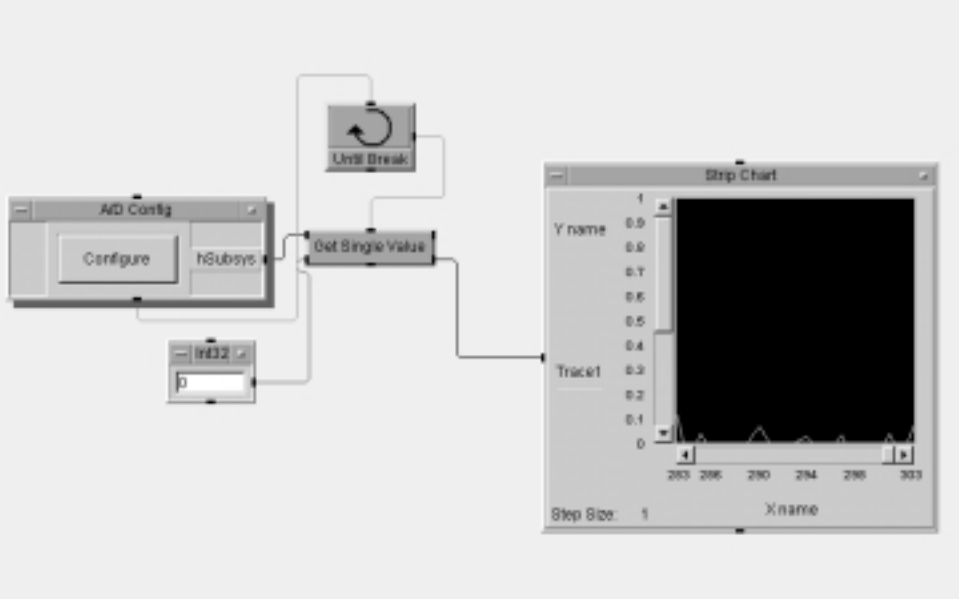

Figure 27: Output when Acquiring a Single Value at a Time

4. Stop the program by clicking the **Stop** button .

Step 5: Save Your Program

Save your program:

1. Click the **Save** button .
2. In the Save File dialog box that appears, type a name such as SINGLE.VEE.
3. Click **Save** to close the dialog box.

Step 6: Collect the Data

Now, we will modify the program, collect the data into an array, and associate the values with time. Because single value acquisition is not clocked by the analog input subsystem's pacer clock, values are acquired at imprecise intervals. You have no way of knowing how much time elapsed between data samples; therefore, it is not a very accurate way of sampling data. When working with signals that change slowly over time (such as the temperature of an oven), the fact that the data is not clocked may not impact your results.

However, you can approach an accurate sampling of the data by associating each value with a constant time interval. Doing this transforms the data into waveform format. A waveform is needed for some display types (the Waveform (Time) object, for example) as well as some of the analysis objects. To collect the data and associate it with time:

1. Select the Strip Chart object and delete it by clicking the **Cut** button on the tool bar .

2. Select the Until Break object and cut it as well.

3. Select **Flow** ⇨ **Repeat** ⇨ **For Count** and place the For Count object in the same location that the Until Break was in.

4. Change the value in the For Count object to 100.

5. Select **Data** ⇨ **Collector** and place the Collector object below and slightly to the right of the Get Single Value object.

6. Double-click the Collector object to open it.

7. Connect the data output pin from the Get Single Value object to the data input pin on the Collector object.

8. Connect the sequence output pin from the For Count object to the XEQ data input pin on the Collector object.

The XEQ (read, “execute”) pin is a “ping” used to cause the Collector object to execute. We want the Collector object to collect when all 100 samples have been acquired. Therefore, when the For Count object is done looping, its sequence output pin activates, causing a “ping” on the Collector object’s XEQ pin.

9. Connect the For Count object as shown in Figure 28.

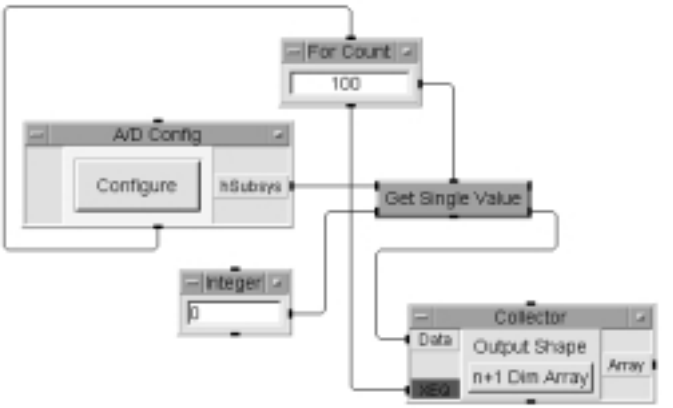

Figure 28: Changing the SINGLE.VEE Program

Step 7: Build a Waveform

You are nearly done. Once it has been collected in the Collector object, the data is stored in a one-dimensional array.To transform this array into a waveform, use a Build Waveform object. This object takes a set of values and distributes the values evenly over a span of time. It then outputs a waveform. To add a Build Waveform object:

1. Select **Data ⇨ Build Data ⇨ Waveform** and place the Build Waveform object to the right of the Collector object.

2. Double-click the Build Waveform object to open it.

3. Connect the data output pin labelled “Array” from the Collector object to the data input pin labelled “Array” on the Build Waveform object.

4. Select **Display ⇨ Waveform (Time)** and place the object to the right of the Get Single Value object.

5. Connect the data output pin from the Build Waveform object to the data input pin on the Waveform (Time) object.

6. Minimize all the objects (except the Waveform (Time) object) by clicking the minimize button in the right-hand corner of each object. You can move objects around to arrange them nicely.

   The program looks something like that shown in Figure 29.

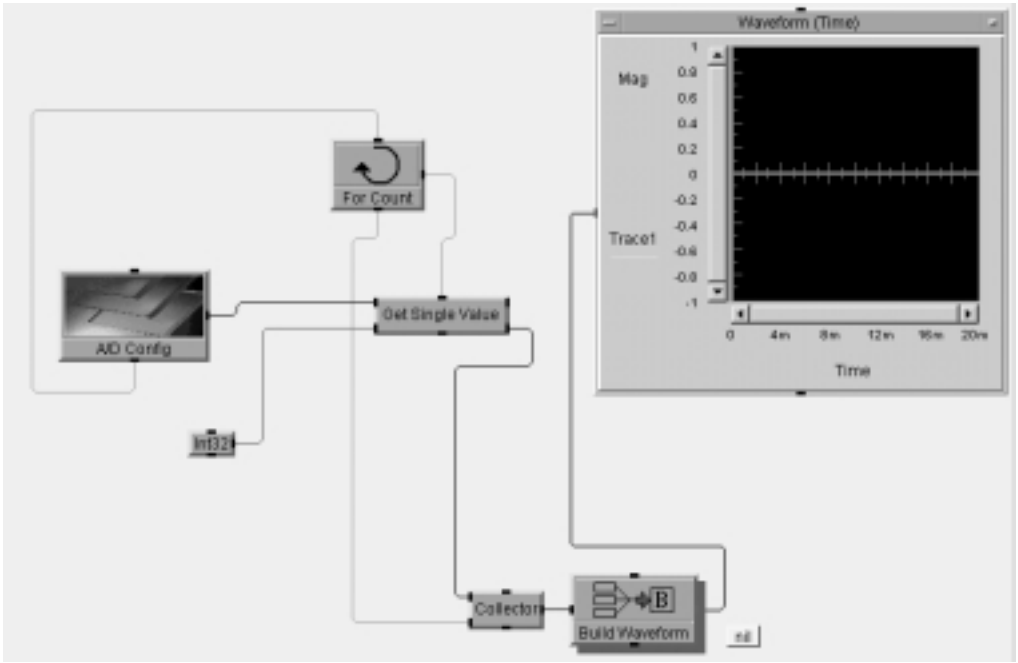

Figure 29: Ready to Build a Waveform

7 minutes

Note: Since we do not know the actual duration over which we are collecting single values, the time span in the Build Waveform object is arbitrary, but will do for this example. The example program ADSINGLE.VEE uses a Timer object to determine the duration and then uses that value as the time span in the Build Waveform object.

Your program is ready to run.

Step 8: Save and Run!

To save and run your program, perform these steps:

1. Save your program by clicking the **Save** button on the tool bar .

2. Use the same settings on the external function generator as you did before (a square wave at 100 Hz).

3. Run the program.
   Acquiring 100 single values takes a few moments, so be patient.

   The output is displayed in the Waveform (Time) object, as shown in Figure 30.

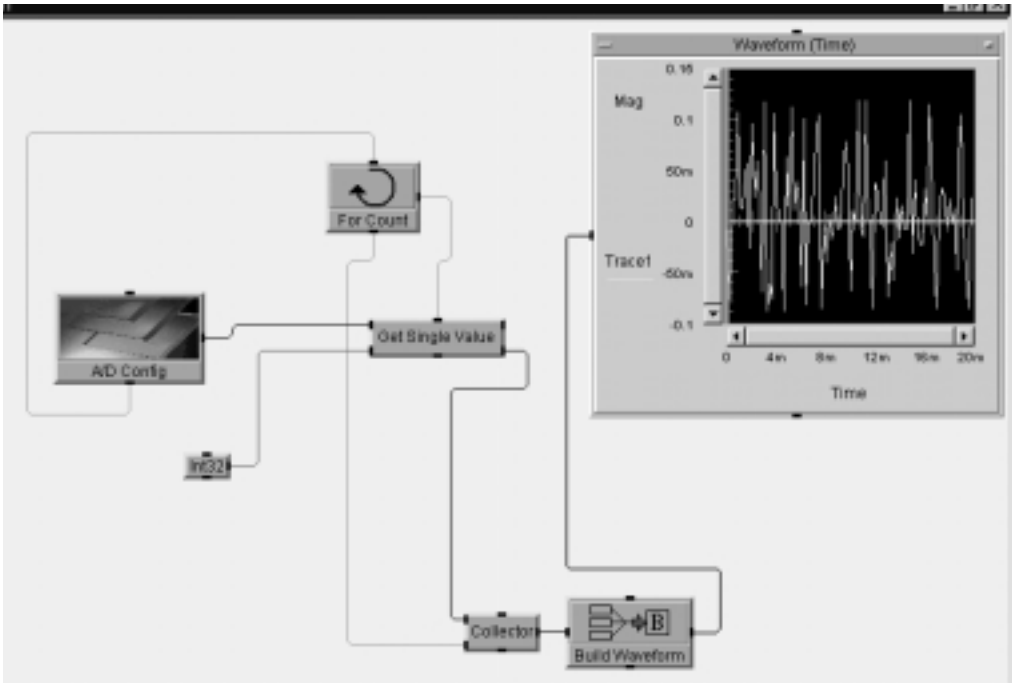

Figure 30: Output after Acquiring Single Values and Building a Waveform

4. Save your program again by clicking the **Save** button on the tool bar .

Lesson 5: Output Analog Data

Digital data can be converted to analog data, and, with the aid of your data acquisition hardware, output to external analog devices such as an oscilloscope. You can use digital data generated internally in VEE, or use acquired analog data that has been converted to digital values by your data acquisition hardware.

In Lesson 5, you will create a new program that generates some data using a Function Generator object, displays the signal in VEE and outputs an analog signal to an oscilloscope, if one is connected.

Restart VEE if you have quit. If VEE is already running, clear the work area by clicking the **New** button on the tool bar .

Note: If your board does not support continuous output, you cannot use the Put Data Panel object when outputting data. You need to use the Put Single Value object instead. Refer to "Outputting Analog Data from a Single Channel," starting on page 156.

To determine if your board supports continuous output, locate the device driver documentation that accompanies your data acquisition hardware. Check the row labelled "Continuous Operation Support," in the Supported Options table. If there is "Yes" in the column under the D/A subsystem, then your board can continuously output data using the Put Data Panel object. Otherwise, you must use the Put Single Value object instead.

Step 1: Gather the Equipment

Before beginning this lesson, make sure you have the equipment needed to perform analog output. For this lesson you need

- An installed data acquisition board from DataTranslation and its Windows device driver (refer to your hardware and device driver documentation).
- A screw terminal panel connected to the data acquisition board in your system (refer to your screw terminal panel documentation).
- (Optional) an external analog device to which the program outputs the signal. Typically this device is an oscilloscope, but depends on your specific needs. If one is available, connect an oscilloscope to D/A channel 0 on the screw terminal panel. The program will still run if you do not have an oscilloscope connected.

Step 2: Add the Objects

To perform analog output, add an object that generates the data (if you are not using acquired data), an object that configures the hardware, and an object that outputs the data. Optionally, you might want to add an object that displays the data:

1. Select **DT DataAcq ⇨ D/A Config** and place the object in the upper middle of the work area.
2. Select **DT DataAcq ⇨ Data ⇨ Put Data Panel** and place this object under the D/A Config object.
3. Select **Display ⇨ Waveform (Time)** and place it under or to the right of the Put Data Panel object.
4. Select **Device ⇨ Virtual Source ⇨ Function Generator** and place the object to the left of the work area.

5. Modify the Function Generator object's Time Span to be 10 m (milliseconds), and its NumPoints to be 1000.

6. Connect the objects as shown in Figure 31.

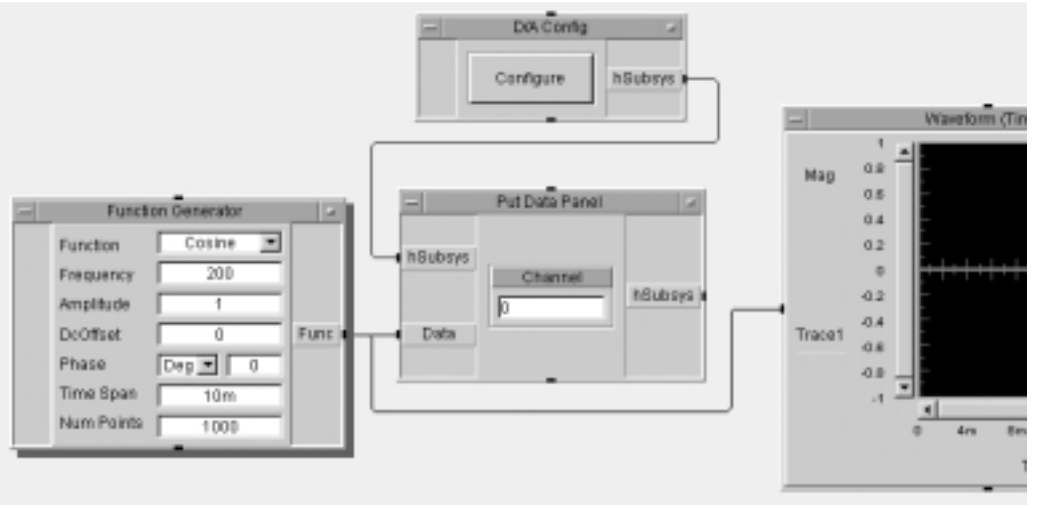

Figure 31: Ready To Output Generated Data

Step 3: Configure the D/A Subsystem

Before you can run the program, you need to configure the D/A subsystem on the data acquisition board:

1. Click the **Configure** button on the D/A Config object.
The D/A Configuration dialog box appears, as shown in Figure 32.

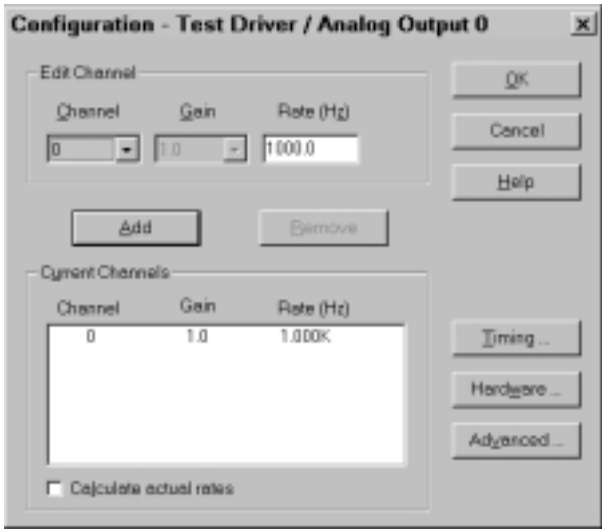

Figure 32: Configuring the D/A Subsystem

Note: If you have more than one data acquisition board installed, a dialog box asks you to indicate which board to use. You can access only one board at a time.

The D/A Configuration dialog box enables you to specify which channels you wish to use for data acquisition, and the sampling rate and gain for each.

2. Change the sampling rate for channel 0 to 2000 Hz.

3. Click **Add** to make the change.

4. Click **OK** to close the dialog box.

Note: Refer to "Selecting an Output Rate," starting on page 150 for more information on setting the D/A sampling rate to match the frequency, time span, and number of points in the Function Generator object.

Your program is ready to run. If you have an oscilloscope connected, make one last check to be sure it is turned on.

Step 4: Run!

Run your program:

1. Click **Auto Scale** on the Waveform (Time) object to see the output better. The output is shown in Figure 33. The waveform is also displayed on the oscilloscope, if one is connected.

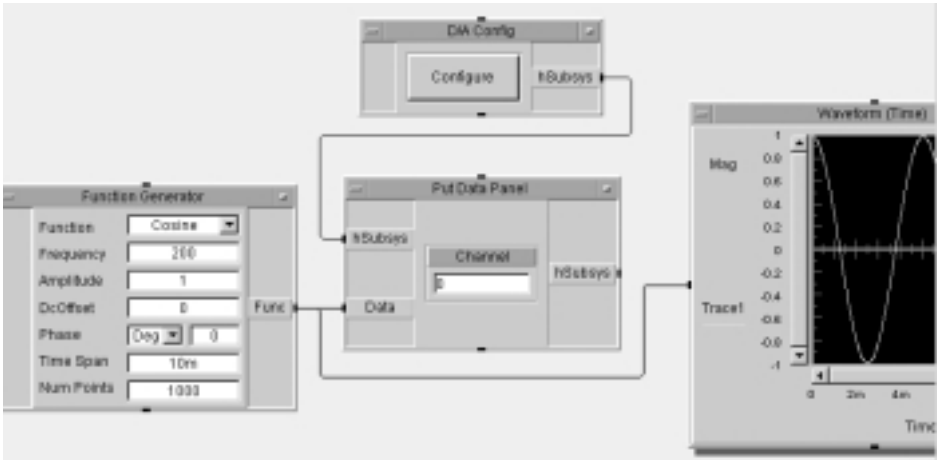

Figure 33: Outputting a Generated Waveform

Step 5: Save Your Program

Though this program will not be used in the next lesson, you might want to save it for your own use later. Save your program:

1. Click the **Save** button on the tool bar .
2. In the Save File dialog box that appears, type a name such as DA.VEE.
3. Click **Save** to close the dialog box.

Lesson 6: Digital Input and Output

Sometimes you need to gather data from a device that is itself digital. Most data acquisition boards from Data Translation provide 16 digital lines, which can be grouped into two *ports* of 8 lines each. Each port can be configured for input or output, or they can be combined to form a single 16-bit port.

In Lesson 6, you will create a new program that acquires digital data from a digital device and displays the signal in VEE.

Restart VEE, if you have quit. If VEE is already running, clear the work area by clicking the **New** button on the tool bar .

Step 1: Gather the Equipment

Before beginning this lesson, make sure you have the equipment needed to perform digital input and output. For this lesson you need

- An installed data acquisition board from DataTranslation and its Windows device driver (refer to your hardware and device driver manuals).
- A screw terminal panel connected to the data acquisition board in your system (refer to your screw terminal panel manual).
- A digital device for input connected to digital port 0 on the screw terminal panel. Make sure the device is turned ON.

Step 2: Add the Objects

To perform digital input, add one object that configures the input port, one object that gathers the data, and one object that displays the data:

1. Select **DT DataAcq** ⇨ **Digital In Config** and place the Digital In Config object in the left of the work area.
   This object configures a digital port for input.

2. Select **DT DataAcq** ⇨ **Data** ⇨ **Get Single Value** and place this object to the right of the Digital In Config object.
   The Get Single Value object acquires one sample from the digital input port immediately upon executing.

3. Select **Data** ⇨ **Constant** ⇨ **Int 32** and place the Int 32 object to the left of the Get Single Value object.
   To execute, the Get Single Value object must have all inputs satisfied. However, when doing digital input or output, the channel number is meaningless. Therefore, this object merely provides an input to the Get Single Value object.

4. Select **Flow** ⇨ **Repeat** ⇨ **Until Break** and place the Until Break object above the Get Single Value object.

5. Select **Display** ⇨ **Logging AlphaNumeric** and place the Logging AlphaNumeric object to the right of the Get Single Value object.

6. Connect the objects as shown in Figure 34.

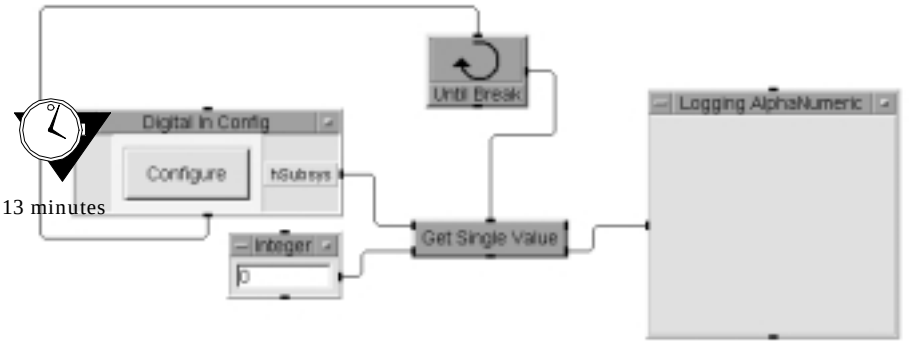

Figure 34: A Simple Program To Acquire Digital Input

Step 3: Configure the Digital Input Port

Before you can run the program, you need to configure the digital input port on the data acquisition board using the Digital Input Config object. If you were performing digital output, you would use the Digital Output Config object. You use one configuration object for each digital port:

1. Click the **Configure** button on the Digital In Config object. *The Select Device and Subsystem dialog box appears.*

2. If you have more than one data acquisition device installed, choose the device you want to use. If that board has more than one digital input port, you can select from the choices. You should select or see a grayed option for **DIN Unit 0** in the Subsystem to Use list box.

3. Click **OK** to close the dialog box and to configure the digital port for input.

Your program is ready to run. Make one last check to be sure your equipment is correctly connected and turned on.

Step 4: Run!

Run your program:

1. The output is displayed in the Logging AlphaNumeric object, as shown in Figure 35. In this example, only bit 0 is being sampled.

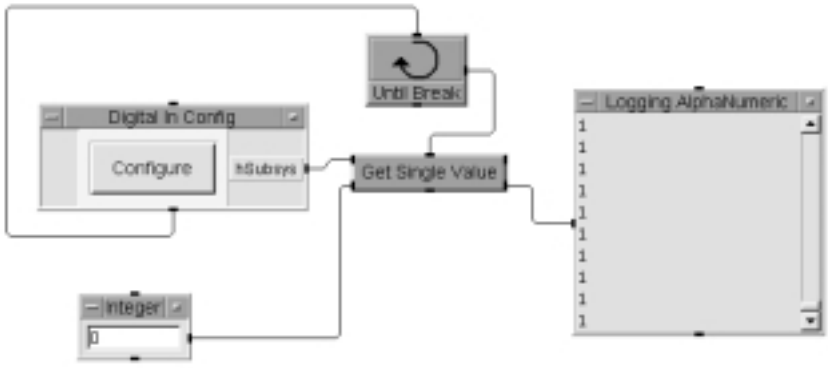

Figure 35: Output from the Digital Input Program

2. Stop the program by clicking the **Stop** button on the tool bar.

Step 5: Save Your Program

Save your program:

1. Click the **Save** button .
2. In the Save File dialog box that appears, type a name such as DI.VEE.
3. Click **Save** to close the dialog box.

Lesson 7: Change the Configuration Programmatically

In Lesson 7 you will modify the simple analog input program you created in Lesson 3 to add the capability of changing the sampling rate by using objects in the program. The objects found in the **Set Config** submenu are handy for changing one aspect of a subsystem's configuration, while keeping all other aspects the same.

Restart VEE, if needed.

Open the program AD.VEE you created in Lesson 3 by clicking the **Open** button on the tool bar . In the Open File dialog box, navigate to the directory \MY DOCUMENTS\VEE programs, and double-click the file AD.VEE.

Note: If your board does not support continuous data acquisition, you cannot do this lesson as written. Lesson 4 covers how to use the Get Single Value object instead of the Get Data Panel object.

To determine if your board supports continuous data acquisition, refer to page 43.

Step 1: Gather the Equipment

Before beginning to write data acquisition programs in VEE, make sure you have the equipment needed to perform analog input. For this lesson you need

- An installed data acquisition board from DataTranslation and its Windows device driver (refer to your hardware and device driver documentation).

- A screw terminal panel connected to the data acquisition board in your system (refer to your screw terminal panel documentation).
- A function generator or other signal source connected to A/D channel 0 on the screw terminal panel. Make sure the signal source is turned ON.

Step 2: Modify the Program

We will attempt this lesson in stages. First, we will make sure the Simple analog input is working. Then we will add the configuration objects, but set them so they do nothing. Then, we will add the objects to change the configuration:

1. Select the Until Break object and delete it by clicking the **Cut** button on the tool bar .
The connections to and from the object are deleted as well. The A/D Config, Get Data Panel, and Waveform (Time) objects remain.

2. Minimize the Get Data Panel object by clicking the button in the right-hand corner of the object.

3. Click the **Configure** button on the A/D Config object.
The A/D Configuration dialog box appears.

The default configuration in the dialog box specifies that channel 0 is sampled at a rate of 1000 Hz (samples per second) and at a gain of 1. These settings are sufficient for our example.

4. Click **OK** to close the dialog box and to configure the A/D subsystem with these settings.

5. Minimize the A/D Config object.

6. Set the external function generator to output a waveform (sine will be fine) at a frequency of 100 Hz. Make sure it is connected to A/D Channel 0 and turned on.

7. Run the program. After clicking the Auto Scale button on the Waveform (Time) object, the output looks something like that shown in Figure 36.

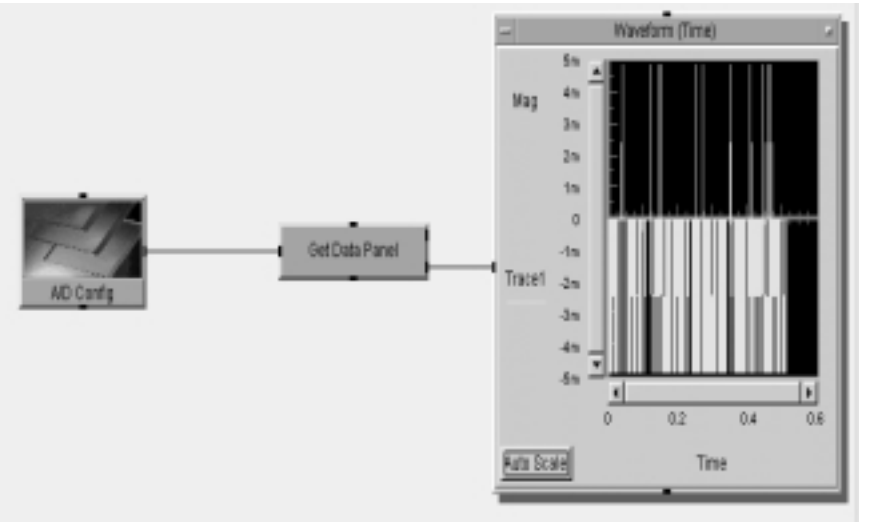

Figure 36: Another Simple Analog Input Program

Step 3: Add the Configuration Objects

The Set Sampling object is a configuration object that enables you to change the sampling rates or gains for all configured channels using objects in the program. As inputs, this object takes the subsystem's handle, the channels, a gain for each channel, and a sampling rate for each channel. It reconfigures the A/D subsystem and outputs an updated subsystem handle (hSubsys).

Often, the Set Sampling object is placed directly after a Get Sampling object in a program. The Get Sampling object finds out the current settings for the channels, gains, and sampling rates. These two objects are often used as a pair so that you can change *one* aspect of configuration, while leaving all other aspects the same.

To add the configuration objects:

1. Select the **Delete Line** button on the tool bar and click the connection between the A/D Config object and the Get Data Panel object.
   The connection is deleted.

2. Select **DT DataAcq** ⇨ **Get Config** ⇨ **Get Sampling** and place the Get Sampling object to the right of the A/D Config object.

3. Select **DT DataAcq** ⇨ **Set Config** ⇨ **Set Sampling** and place the Set Sampling object to the right of the Get Sampling object.

4. Connect the objects as shown in Figure 37.
   At this stage of the example, all outputs from the Get Sampling object are connected to the Set Sampling object. No configuration aspects are being changed. Later, you will add a user input slider that will adjust the sampling rate.

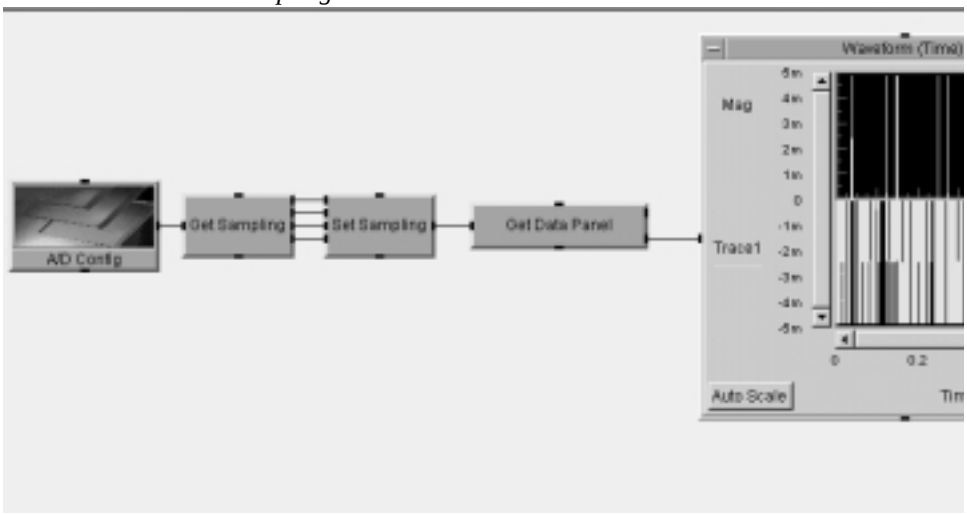

Figure 37: Adding Configuration Objects

5. Just to make sure everything is still working, run your program again. The output should look the same as that shown in Figure 37.

Step 4: Add User Input

At this point you are ready to add the user input object:

1. Select the **Delete Line** button on the tool bar and click the lowest connection (the fourth connection from the top) between the Get Sampling object and the Set Sampling object.

   This connection is for the sampling rate. To verify this, open the Set Sampling object (by double-clicking it) and notice the label "Rates" on the lowest data input pin (see Figure 38). The object expects a one-dimensional array of sampling rates, one for each configured channel. In this lesson, we are using only one channel, so we are going to create a one-dimensional array containing one value—the new sampling rate.

Figure 38: The Set Sampling Object

2. Select **Data** ⇨ **Continuous** ⇨ **Real 64 Slider** and place the Real Slider object below the A/D Config object.

3. Change the lower limit to **200** and the upper limit to **5000**. These value are in Hz.

It is necessary to edit the limits on the Real Slider object, the lower one especially, because the value output from the Real Slider object is going to be the new sampling rate for analog data acquisition. (The previous sampling rate was set in the A/D Configuration dialog box.) To avoid *aliasing*, you must sample a signal at least twice as fast as the highest frequency component in the signal. Since our signal is a 100 Hz sine wave, then sampling at 200 Hz is the minimum sampling frequency we can use to avoid aliasing errors. (For more information on aliasing, see "Selecting a Sampling Rate," starting on page 84.)

4. Set the Real Slider to 1000. (Enter **1000** in the top-most edit box.)
   This setting mimics the sampling rate you configured in the A/D Configuration dialog box.

5. Select **Data** ⇨ **Allocate Array** ⇨ **Real** 64 and place the Alloc Real object to the right of the Real Slider object.

6. Click the text "Lin Ramp" to open a drop-down list and select **Init Value** from the list.

7. Change the Size to **1**.

8. With your cursor over the object, click the right mouse button to invoke the object menu, and select **Add Terminal** ⇨ **Data Input**.

9. From the small dialog box that appears, select **Init Value** and click **OK**.
   A new data input pin is added to the Alloc Real object.

10. Connect the Real Slider object to the Alloc Real object.

11. Connect the Alloc Real object to the available data input on the Set Sampling object.
    The program looks like that shown in Figure 39.

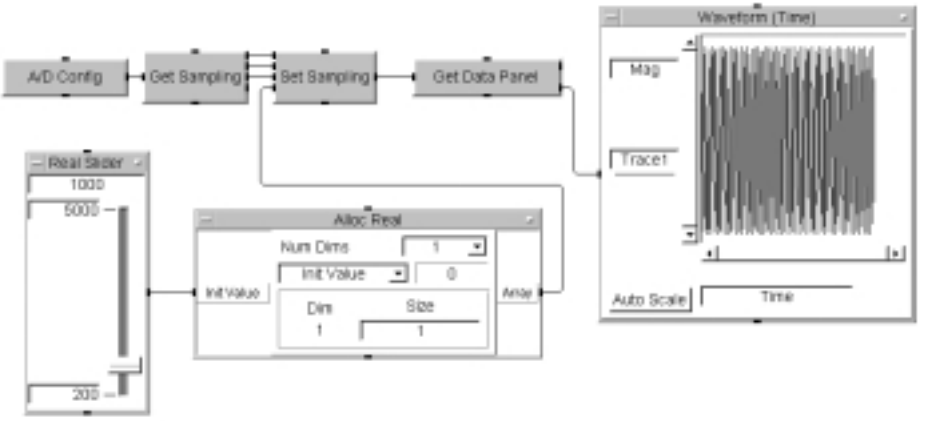

Figure 39: All Objects Are in Place

Your program is ready to run. Make one last check to be sure your equipment is correctly connected and turned on.

Step 5: Run!

Run your program:

1. When acquisition stops, change the setting on the Real Slider object by dragging the knob up or down. Note the effect in the Waveform (Time) object each time you run. Figure 40 shows the output with the sampling rate set to 3961 Hz.

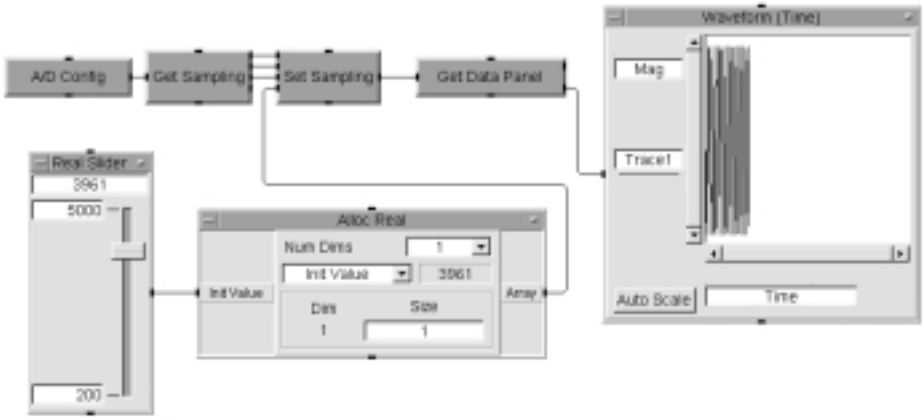

Figure 40: Changing the Sampling Rate with a Real Slider

Step 6: Save Your Program

If you want to save the changes you just made to the original AD.VEE program, click the Save button . Otherwise, select Save As from the File menu and provide a different file name, such as AD2.

What's Next?

Now that you have finished this introduction to DT VPI's data acquisition capabilities, try some things on your own. Use the following documents in addition to this one to help you:

- VEE Multimedia tutorials,
- VEE on-line help,
- *VEE Pro User's Guide,*
- *VEE Pro Advanced Techniques,*
- *VEE OneLab User's Guide,*
- *VEE OneLab Advanced Techniques,* and
- DT VPI on-line help.

Analog Input

Connecting External Sensors 81
Configuring for Analog Input 81
Starting an Analog Input Acquisition 91
Acquiring Data from a Single Channel 92
Acquiring Data from Multiple Channels 94
Acquiring Analog Data Continuously 96
Acquiring a Single Analog Value 101
Acquiring a Burst of Data 104
Acquiring Pre-Trigger Data 109
Acquiring Data Using a Triggered Scan 112
Changing the Subsystem Configuration 117
Handling Acquisition Errors 121
Sending Data to a Disk File 125
Sending Data Through DDE 139
Optimizing Analog Input 142

This chapter discusses how to control your data acquisition hardware to perform analog input. It includes information on configuring the channel list, setting the sampling rate and gain, and using an external clock or trigger.

This chapter also includes many example programs and explanations of using the data acquisition objects provided by D TVPI.You will learn how to create programs to

- Acquire data and display it;
- Acquire data on several channels;
- Acquire data continuously;
- Acquire a single value repeatedly;
- Acquire a known number of samples or acquire for a known duration;
- Acquire data before a trigger event;
- Acquire scans of data when a trigger occurs;
- Handle errors; and
- Send data to a file or to another application through DDE.

Connecting External Sensors

Before you can acquire analog data, you need to connect your external sensor or measuring device. See the hardware manual for your board for information on connecting external devices.

Configuring for Analog Input

Now that you have connected the external devices that provide your VEE program with analog input, it is time to start building your data acquisition program. Every VEE program that includes data acquisition needs a *subsystem configuration object*. The first five menu items in the **DT DataAcq** menu access subsystem configuration objects. For an analog input program, you need the A/D Config object. Select **DT DataAcq ⇨ A/D Config,** and click the **Configure** button.

Table 2 directs you to the appropriate dialog box to perform various configurations. These configurations are discussed in the following pages.

Table 2: Configuring the A/D Subsystem

To Configure...	In the A/D Config Object, click...
The A/D channel list	**Configure**
Sampling rates and gains for channels	**Configure**
Sampling rates, gains, filters, burst mode, write to file, and data mode (Gap Free or Allow Gaps) for channels	**Configure**, then **Advanced**
The input voltage range	**Configure**, then **Hardware**
The input channel type	**Configure**, then **Hardware**
The clock source	**Configure**, then **Timing**
The trigger source and triggering mode	**Configure**, then **Timing**

Selecting a Board when More than One Data Acquisition Board Is Installed

If you have more than one data acquisition board installed in your computer, a dialog box similar to that shown in Figure 41 appears asking you to specify the board to use.

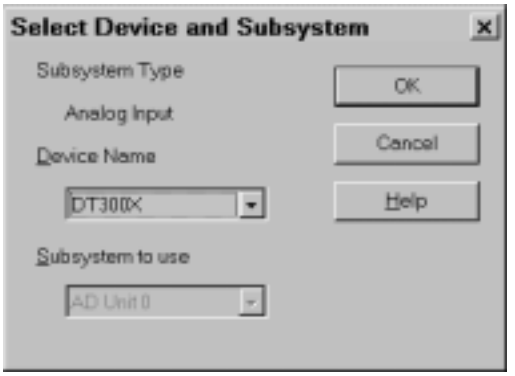

Figure 41: Choosing One of Several Installed Devices

Note: This dialog box appears only if you have more than one data acquisition board installed in your system or more than one subsystem unit on that board.

Select the name of the board you want to use from the Device Name drop-down list, and then select **AD Unit 0** from the Subsystem To Use drop-down list, and click **OK**.

Note: The name of the board is the name assigned to it when its device driver was installed. Select the board that has the capabilities you need for analog input.

Building a Channel List

The A/D Configuration dialog box, shown in Figure 42, is used to build a *channel list*, which specifies which channels you intend to use for analog input, and the sampling rate and gain for each channel.

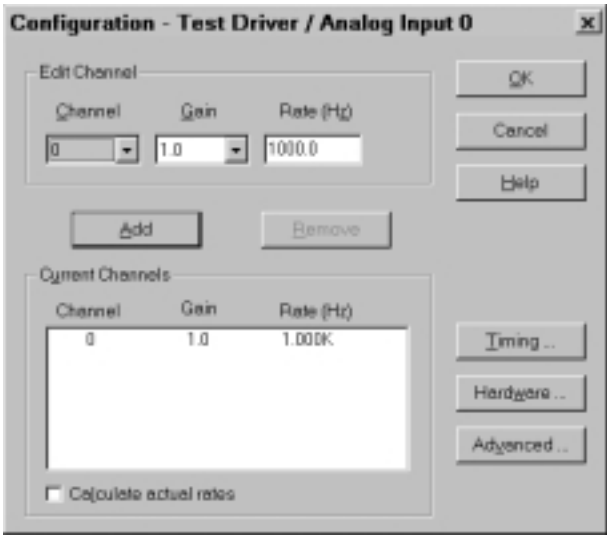

Figure 42: The A/D Configuration Dialog Box

Take a moment to consider the external connections you have made. For each channel you intend to use, enter the gain in the Gain edit box, and the sampling rate needed in the Rate (Hz) edit box, then click **Add**. The channel number, gain, and associated rate are added to the channel list in the Current Channels box. If you need to edit one of the channels, click on the entry in the Current Channels box; the information for that channel appears in the Edit Channels box.

The Calculate Actual Rates checkbox refers to the actual rates that the analog input subsystem can achieve. When this box is checked, the Current Channels box reflects the actual sampling rates. To view the sampling rate that you originally entered, select the channel in the Current Channels box; the original rate for that channel is displayed in the Edit Channel box.

By default, the A/D Configuration dialog box shows channel 0 in the channel list. You can add as many channels to the channel list as your board can accommodate, and you can remove all but one channel. Your channel list is always sorted by channel number, starting with the lowest number. It is unnecessary to define the channel numbers sequentially, as in channel 0, then 1, then 2 and so on.You should note, however, that when you run the program, DT VPI acquires data on the channels in the order that they appear in the channel list.

Selecting a Sampling Rate

According to sampling theory (the Nyquist theorem), a data acquisition system should sample an analog input signal at least twice as quickly as the input's highest frequency component. For example, to accurately sample a 20 kHz signal, the sampling rate should be at least 40 kHz. Doing so avoids an error condition called *aliasing*, in which high frequency input components erroneously appear as lower frequencies after sampling (see Figure 43). Often the erroneous signal is indistinguishable from a correctly sampled signal. Some data acquisition boards have anti-aliasing filters built right into them.

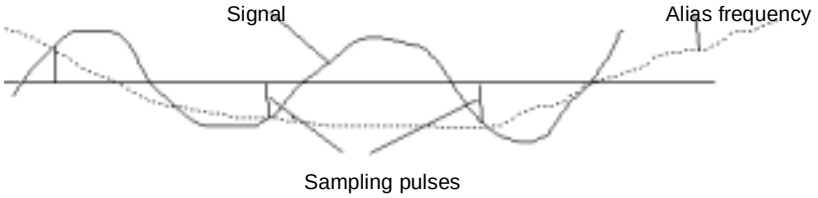

Figure 43: An Aliased Frequency as a Result of Sampling Pulses Too Far Apart

Filtering an Analog Input Signal

If your board has anti-aliasing filters, click **Advanced**. The dialog box shown in Figure 44 appears; you can use this dialog box to specify more details about the analog input configuration. In the Filter (Hz) edit box, select from the available hardware low-pass filter cutoff frequencies.

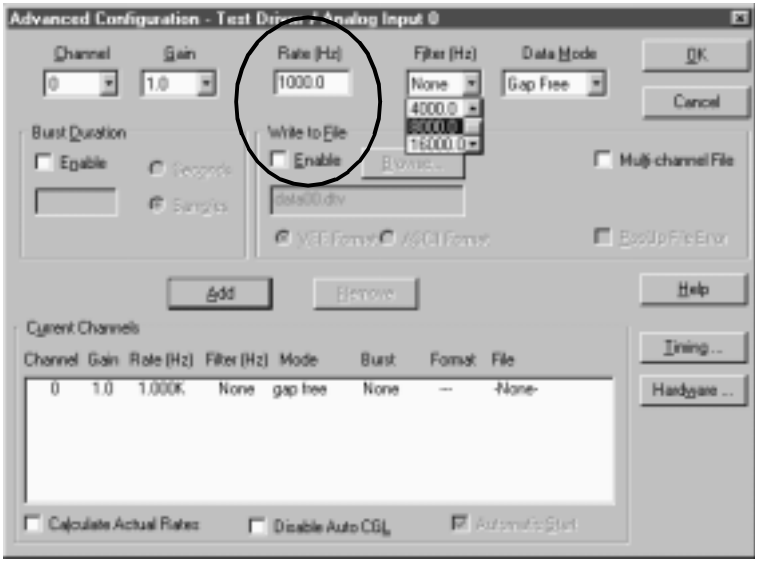

Figure 44: The A/D Advanced Configuration Dialog Box

Selecting the Gain

Some data acquisition boards provide several input ranges by using software programmable gain amplifiers, or *gain*. When you considered which data acquisition board to use with your VEE programs, one of the considerations was the range of the analog input signals you would be working with. You should have chosen a board that most fully covers the analog input range you are interested in measuring. Doing so gives you the greatest number of samples in that range, and thus the greatest accuracy.

Depending on the capabilities of your data acquisition hardware, you can select a gain of 1, 2, 4, or 8, or 1, 10, 100, or 500. Use the following equation to determine which gain to use.

$$\frac{HardwareRangeSetting}{Gain} = RangeOfInputSignal$$

The ranges for *HardwareRangeSetting* are board-specific. Consult your hardware user manual or check the Hardware dialog box (click **Hardware**) to see the ranges available for your board's current configuration.

For example, suppose you selected a range of ±10 V from the available choices in the Hardware dialog box. If you select a gain of 8, the range of the input signal can be as small as ±1.25 V (since 10 divided by 8 equals 1.25) while still maintaining full sampling accuracy.

If you want a gain other than 1, select a gain from the drop-down list box. To add the new gain value to the channel list, click **Add**.

Selecting an Input Range and Type

To select the input range, click the **Hardware** button from the A/D Configuration dialog box. The Hardware dialog box (shown in Figure 45) gives you the opportunity to select an input signal range from those available on your board. Use this range with the gain setting in the A/D Configuration dialog box to most closely match the actual range of your input signal.

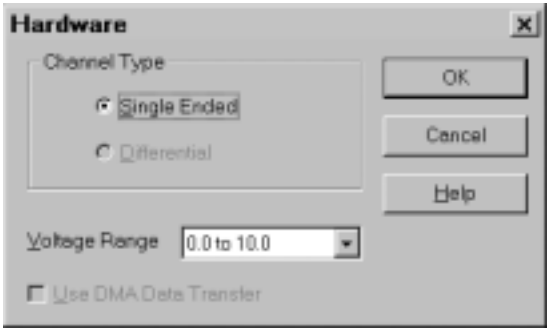

Figure 45: Indicating an Input Signal Voltage Range

Also, select either the single-ended or differential channel type. Your selection should match the actual type of input channel you intend to use. Note, for pseudo-differential wiring configuration, select the single-ended channel type.

Note: For information on the checkbox labelled “Use DMA Data Transfer,” see page 201.

Selecting a Clock and Trigger Source

To specify that you are supplying an external clock to be used as the pacer clock for analog-to-digital conversion, click the **Timing** button from the A / D Configuration dialog box.

Select **External** from the drop-down list under Clock Source. Enter the frequency of the external clock signal (or your closest estimate) in the External Clock Frequency (Hz) box. This frequency must be greater than or equal to the sampling rate for the channels in the Channel List.

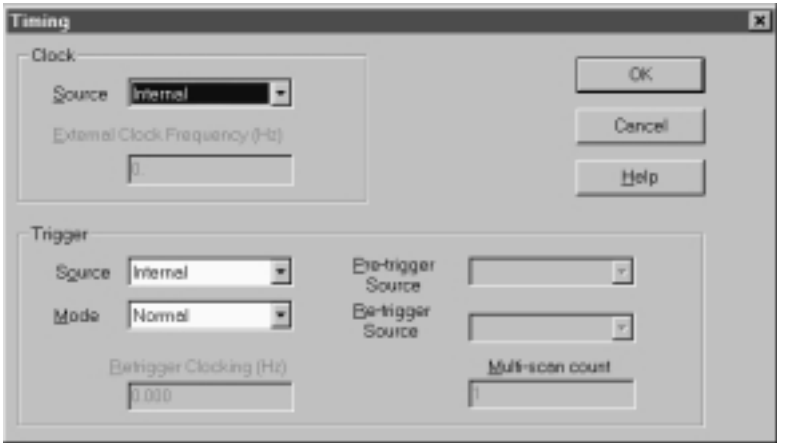

Figure 46: Selecting an External Clock or Trigger

Note that a user counter / timer can be used to generate an external clock signal. See "Using the Counter / Timer as a Clock," starting on page 209.

To specify that you are supplying an external trigger to start the pacer clock, select **External** from the drop-down list under Trigger Source. Leave the Mode as Normal. Click **OK** to close the Timing dialog box.

Specifying a Subsystem Handle

When you are done configuring the analog input subsystem, click **OK**. A configured subsystem has an identifier, called a *subsystem handle*, which is used by other DT VPI objects to identify which subsystem on the hardware to use. The data acquisition objects need information about what channels, gains, and sampling rates to use. All data acquisition objects require a subsystem handle (hSubsys) as one of the data inputs.

To explore the concept of subsystem handle, clear the work area and place an A/D Config object (**DT DataAcq ⇨ A/D Config**) on your work area. Double-click the output terminal **hSubsys** to get information about it. The information dialog box indicates that no data container has been created on the output terminal hSubsys, as shown in Figure 47. Click **OK** to close the dialog box.

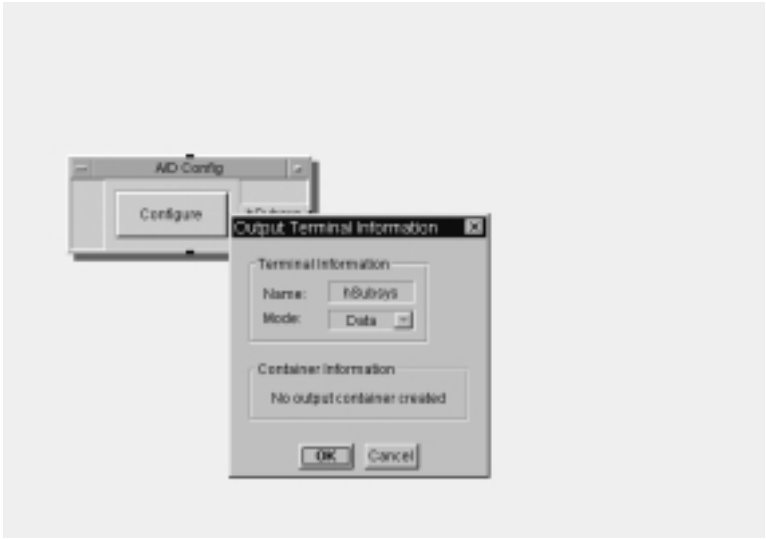

Figure 47: The Output Terminal on the A/D Config Object before It Is Run

Now, click **Run**. If asked, select the data acquisition board you want to use, then click **OK** in the A/D Configuration dialog box to use the default channel list. Double-click the output terminal on the A/D Config object again. Now that the analog input subsystem has been configured, a subsystem handle exists.You should see some value in the data container on the output terminal, as shown in Figure 48.

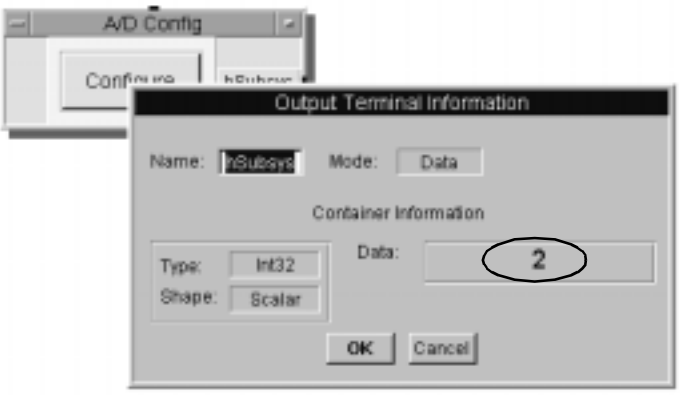

Figure 48: The Output Terminal on the A/D Config Object after It Is Run

Starting an Analog Input Acquisition

There are several ways to start analog data acquisition, and the easiest way is to do nothing! By default, the A/D Config object is set to start acquisition automatically when a Get Data Panel or Get Data Function object executes.

Another way to start acquisition on the hardware is to use the Start Acq object. This object is different than the VEE Start object in that it starts a subsystem on the data acquisition hardware, and does not start the execution of a VEE thread. If you do not want analog input to start automatically when the Get Data Panel or Get Data Function executes, include a Start Acq object before the Get Data Panel or Get Data Function object. The Get Data Panel and Get Data Function objects start acquisition automatically only the first time. If you stop acquisition and want to start it again, you need a Start Acq object.

Acquiring Data from a Single Channel

The simplest data acquisition program acquires a specified number of samples from a single channel. DT VPI example program ADSIMPLE.VEE (in the \Examples\Dataacq\ directory) is shown in Figure 49.

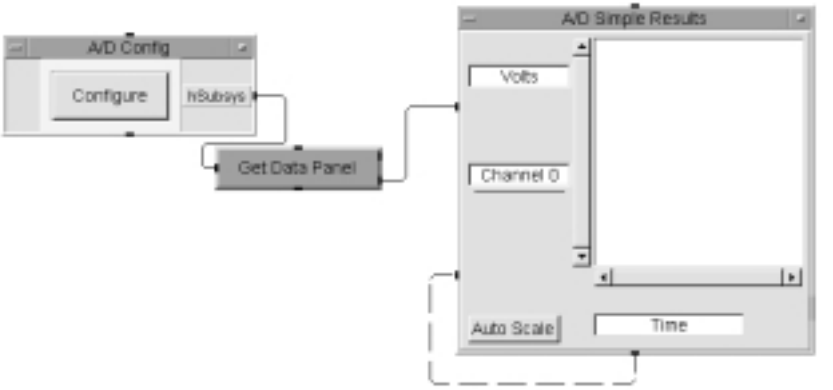

Figure 49: Example on Disk: ADSIMPLE.VEE

As you learned in the last section, the A/D Config object is required. Click the **Configure** button, (select the board to use, if necessary), and specify the channel, gain, and sampling rate configuration. In this example, the default configuration of channel 0 at a sampling rate of 1000 Hz is used.

The Get Data Panel object needs the subsystem handle from the A/D Config object. You must also indicate which channel you have selected in the A/D Configuration dialog box and the number of samples, or data points, to acquire. Enter information directly into the user edit boxes. In this example, we are using the default settings: channel 0 and 1000 samples, or points.

Note: Make sure that each Get Data Panel object has a corresponding channel configured in the A/D Config object. Mismatched channels result in a run-time error.

A display object is often used to display the acquired data. Because the Get Data Panel object outputs waveform data, either a Waveform (Time) object or an XY Trace object can be used. Waveform data has relative time values associated with each data point, and thus can be plotted on a Volts vs. Time graph.

Before running this program, connect the external device that is supplying an analog signal. In this example, a 1 Hz sine wave was used as the input signal. Click the **Run** button on the tool bar. The output is shown in Figure 50.

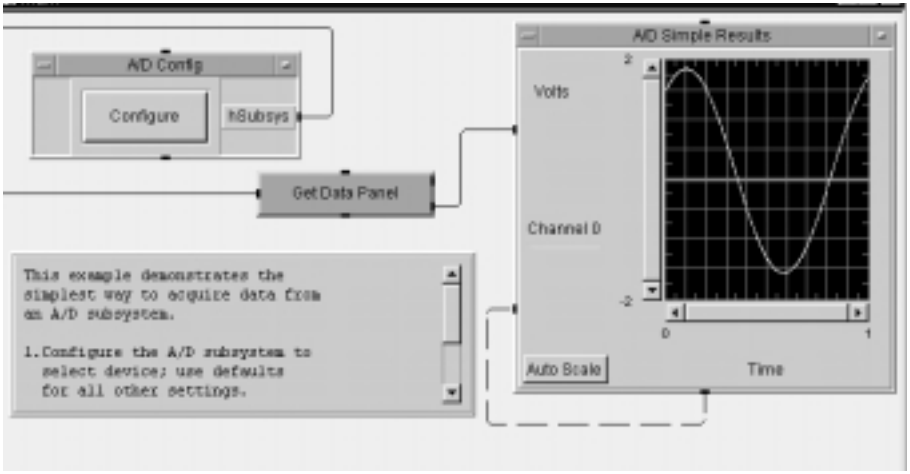

Figure 50: ADSIMPLE.VEE after Acquiring Analog Data

Try other displays, such as Logging AlphaNumeric, Strip Chart, or Spectrum (Freq).

Acquiring Data from Multiple Channels

Acquiring data from several channels is a simple extension of the last program. You need a Get Data Panel object for each channel you select in the A/D Configuration dialog box. For example, Figure 51 shows a program that acquires data on three channels.

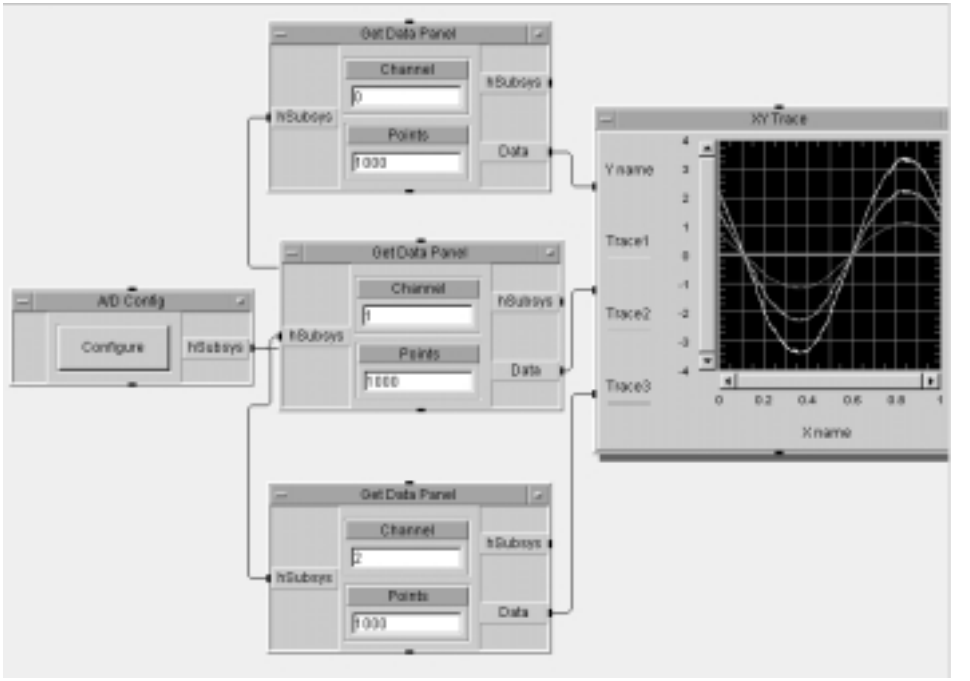

Figure 51: A Program To Acquire Data on Three Channels

A one-to-one correspondence exists between the number of Get Data Panel objects and the number of channels selected in the A/D Configuration dialog box. For this example, the A/D Config object was configured for channels 0, 1, and 2, as shown in Figure 52.

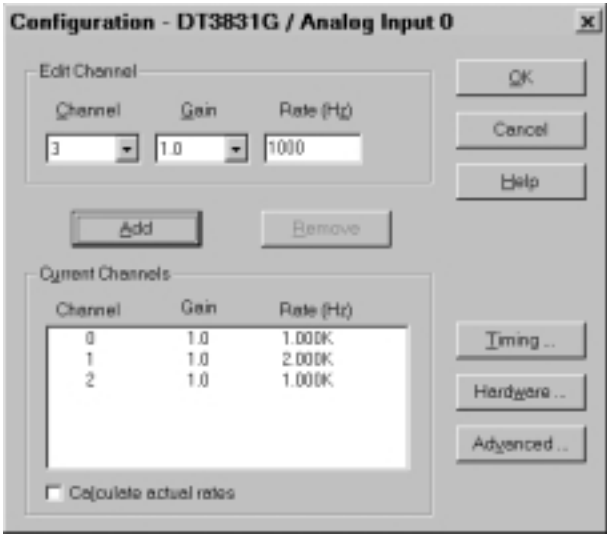

Figure 52: Configuration for Three Channels

Example on Disk: The MULTI.VEE example (in the \Examples\Dataacq\ directory) shows how to acquire data on multiple channels.

Acquiring Analog Data Continuously

To accomplish continuous analog data acquisition, add a flow object to the program we looked at in the last section. This example adds an Until Break object. If you know the exact number of samples you want to acquire or the exact duration over which you want to acquire, *burst mode* is more appropriate. Burst mode is discussed a little later in this chapter.

The program looks something like that shown in Figure 53.

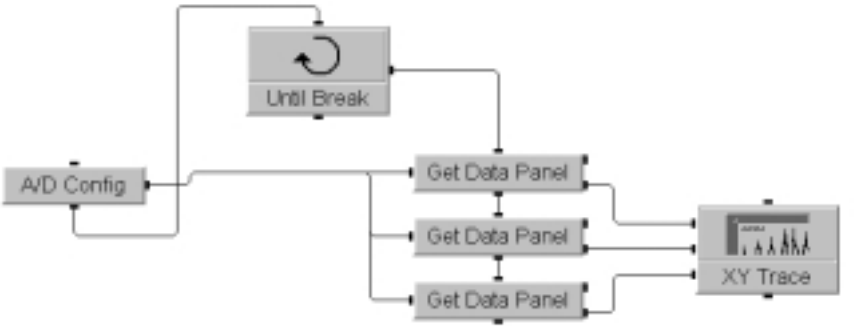

Figure 53: A ProgramTo Acquire Data Continuously

As usual, you need to configure the A/D Config object, setting up your channel list for as many channels as you need. For example, the configuration for the program above is shown in Figure 54.

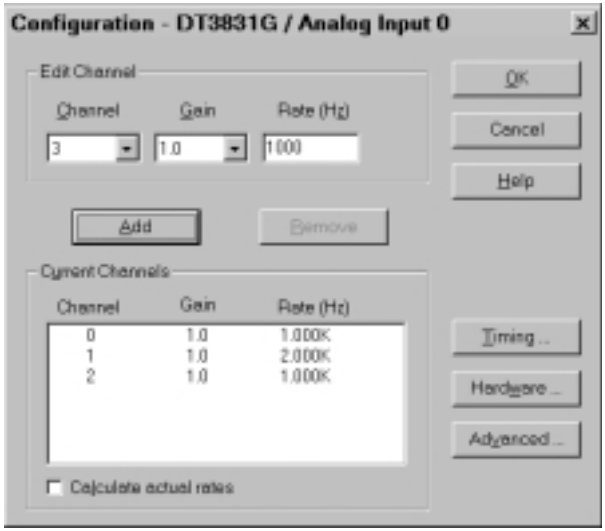

Figure 54: Configuration for Continuous Acquisition on Three Channels

When you run this program, the A/D Config object is executed first. When it is finished, the A/D Config object pings its sequence out pin, which is connected to the sequence in pin of the Until Break object. This connection ensures that the loop starts after the analog input subsystem is configured and has a subsystem handle. Note that the subsystem handle is sent to each of the Get Data Panel objects as input.

On each cycle, the Until Break object pings the sequence in pin of the Get Data Panel object for channel 0, which reads the specified number of samples and then alerts the next Get Data Panel object that it is OK to do so as well (by a sequence ping). When all of the Get Data Panel objects have executed and the data has been graphed, the loop repeats until you stop the program by clicking the **Stop** button on the tool bar. The subsystem continues acquiring data for a few moments more, until the data queues and Mixed Data Buffer is filled.

What's Really Going On?

Figure 55 illustrates how data is stored in memory and then retrieved by DT VPI.

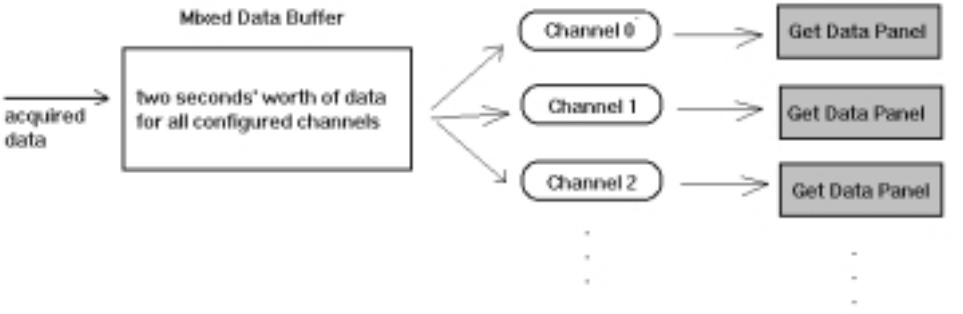

Figure 55: Acquired Data Stored in Memory, then Retrieved by DT VPI

Now, let's look deep into the program to learn how DT VPI handles data:

- When you click **Run**, the A/D Config object initializes the board and sets up the channel list. Enough memory for two seconds of data for all channels is set aside as the *Mixed Data Buffer*. The subsystem is in Ready state.
- When the Get Data Panel object is executed, acquisition begins automatically; the subsystem is running. All three channels are sampled, and the data is stored in the Mixed Data Buffer. At this point, the data for all three channels is still mixed together.

- In between the execution of other objects, the data is pulled out of the Mixed Data Buffer and sorted by channel. Each configured channel has its own memory buffer, called a *data queue*. Enough memory is set aside for each data queue to hold four seconds of data, depending on the channel's sampling rate. As much data as can fit in the data queues is pulled out of the Mixed Data Buffer and sorted into the appropriate data queue. At the same time, more data is acquired and stored in the Mixed Data Buffer, so the process is ongoing.

  The goal is to keep the Mixed Data Buffer as empty as possible. If, however, VEE is executing an object that takes a while (an FFT(x) object, for example), the Mixed Data Buffer might get filled. If this happens, data acquisition stops on all channels and the subsystem is done.

- When enough data is in a channel's data queue, the Get Data Panel object transfers the specified number of samples from memory to VEE. The Get Data Panel object passes the data through its output pin to the display object, where it is graphed. If, for some reason, the data in a data queue is not removed in time, the data queue fills up and begins backing up in the Mixed Data Buffer. If this happens, that particular channel stops (in Gap-Free data mode), or throws out data (in Allow Gaps mode), but the subsystem continues to run.
- When the thread finishes executing, the program stops but data acquisition continues on the hardware until the Mixed Data Buffer fills. Then, data acquisition stops. If you pause or stop the VEE program, data acquisition continues until the data queues and Mixed Data Buffer fill or until a Stop Acq object is executed.

States of an A/D Channel

Since the Get Data Panel returns when it has retrieved all of the samples specified, you may be interested in finding out how many samples still remain in that channel's data queue by executing an Acq Status object. Acq Status also provides information about the status for the channel, as shown in Table 3.

Table 3: States of an A/D Channel

Status from Acq Status Object	Channel Status
Ready	Subsystem is initialized but not started. No data exists in memory yet. Executing a Stop Acq or Set Config object returns each channel in the subsystem to Ready.
Running	The hardware is acquiring data on the channel or awaiting an external trigger. Note that the channel continues to run even after Get Data Panel retrieves its specified samples.
Done	The channel Stopped for one of the following reasons: (a) Burst mode finished. (b)The data mode is Gap-Free and the channel's data queue ran out of space. (c) Hardware errors such as overrun (#2008) or trigger (#2006) errors occurred. Note that the Get Data Panel object can still retrieve all valid data from the data queue, even after the channel is "Done." Error 2012 "Acquisition is Complete for the Channel" is reported only if insufficient data exists to meet the request.

Acquiring a Single Analog Value

Typically, acquiring a single value of analog data is used when you are not concerned with a pacer clock. The Get Single Value object is used to acquire one value immediately when executed. It does not wait for a pacer clock tick, and it does not use the Mixed Data Buffer or a data queue.

If you need to acquire a single analog value, why use a Get Single Value object instead of a Get Data Panel object with a setting of one point? The answer lies in the way the sample is acquired. When the Get Data Panel object is executed, the board acquires data and begins filling the Mixed Data Buffer that was set aside. The data is then passed to the channel's data queue. If you set the Get Data Panel object to retrieve only one sample from the data queue, memory is wasted memory needlessly.

Another reason to use the Get Single Value object instead of Get Data Panel is if your board does not support continuous data acquisition. To determine if your board supports continuous acquisition, check the row labelled "Continuous Operation Support" in the supported options table in the board's device driver documentation. A "Yes" in the column under the A/D subsystem indicates your board can continuously acquire data using the Get Data Panel object. If your board does not support continuous operations, you must use the Get Single Value object.

Suppose you were testing the output of a battery, the temperature of an oven, or some other constant or slowly changing signal. Acquiring a single value is a simple and appropriate method for these situations. Figure 56 shows a typical program using Get Single Value.

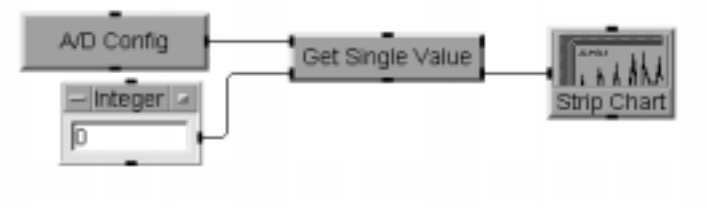

Figure 56: A Program Using Get Single Value

As usual, the A/D Config object configures the analog input subsystem on the data acquisition hardware. However, the Get Single Value object does not refer to the channel list defined in the A/D Configuration dialog box. You need only specify the channel number as an input to the Get Single Value object (usually in the form of an Int 32 object).

The subsystem handle is passed to a Get Single Value object. In addition to the subsystem handle, the Get Single Value object needs to know which channel to access. An Int 32 object provides this information. When you run the program, the Get Single Value object *immediately* acquires one value of the analog signal on the specified channel using the appropriate gain, automatically. It passes the value to the display object.

Notes: The Get Single Value object uses the hSubsys from the A/D Config object, but does not use the channel list. The Int 32 object passes the channel number; the appropriate gain is automatically used to acquire the signal.

Acquiring a single value can be performed only when the channel is in the Ready state.

Example on Disk: The ADSINGLE.VEE example program (in the \Examples\Dataacq\ directory) shows how the Get Single Value object is used with a For Count object (labelled Select # of Samples in this example) to acquire a certain number of samples. The data is passed to a Collector object, is converted into a waveform in a Build Waveform object, and finally displayed. The Timer object times how long it takes from the start of acquisition to finish.

Acquiring a Burst of Data

If you know exactly how many samples of data you want to acquire or the time period over which you want to acquire, use *burst mode.* For example, suppose you want to measure several aspects of a hammer blow on an object. The hammer blows occur once every second. You wish to measure the effect with accelerometers connected to channels 0, 1, and 2. You have obtained the proper accelerometers and made the proper input connections. Figure 57 shows the program.

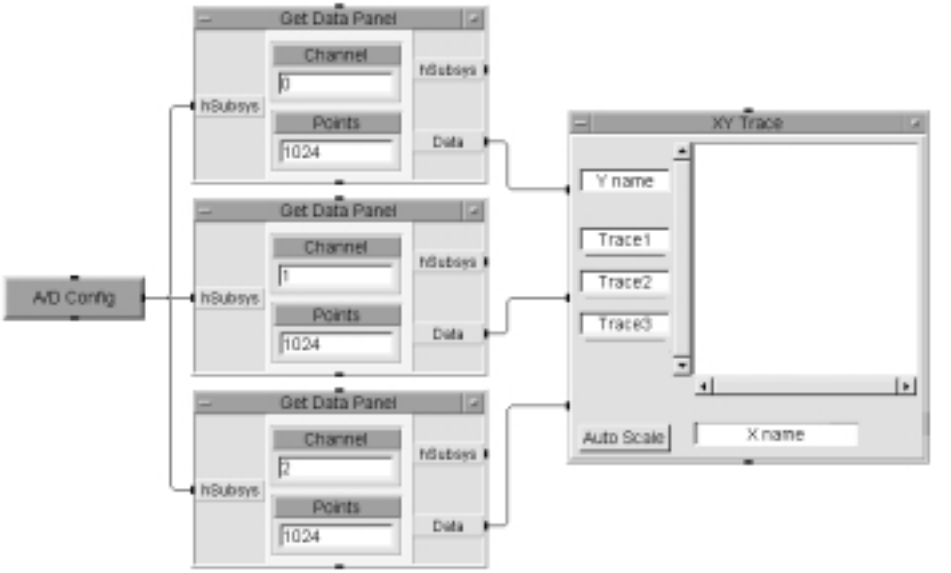

Figure 57: A Program To Acquire Data in Burst Mode

Example on Disk: The example program ADBURST.VEE (in the \Examples\Dataacq\ directory) is a similar program, using only one channel configured for 512 samples in burst mode.

Configuring for Burst Mode

You probably noticed that the program to acquire data in burst mode uses the same objects we have already worked with. The difference lies in the configuration. To set up burst mode, you need to use the A/D Advanced Configuration dialog box. On the A/D Config object, click **Configure**. Because you intend to acquire a burst of data, click **Advanced**. The A/D Configuration dialog box enlarges to show advanced features, one of which is burst mode.

Using the A/D Advanced Configuration dialog box, you can set up the channel list just as you would normally. For our example, we intend to use channels 0, 1, and 2. As you add each channel, click the checkbox labeled **Enable** in the Burst Duration section, and click the radio button next to **Samples**. Set the channels to the following rates and gains:

Channel	Rate (Hz)	Gain	Burst Enabled?	Burst Duration (samples)
0	10000	1	yes	1024
1	10000	1	yes	1024
2	10000	1	yes	1024

Figure 58 shows how the dialog box should look when you are done.

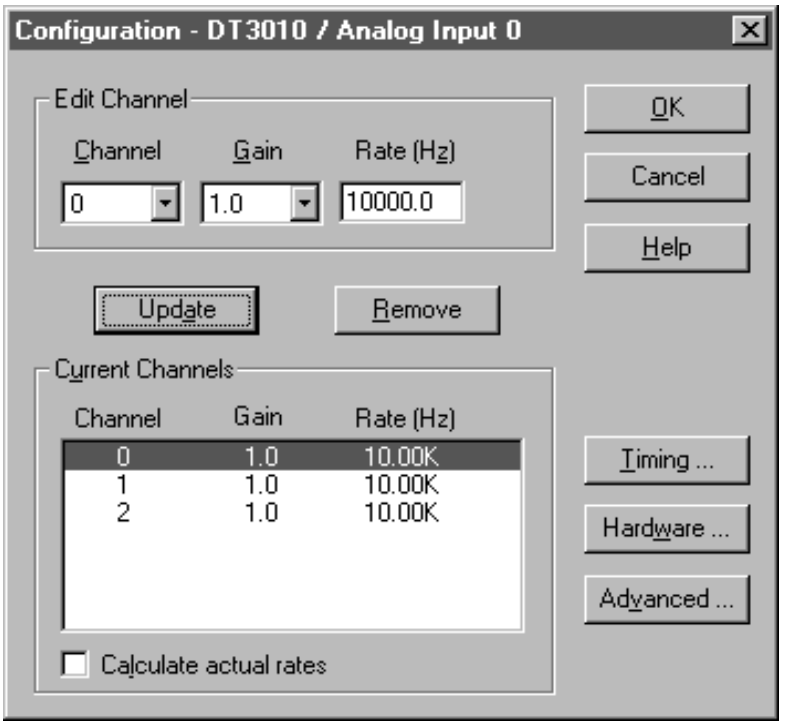

Figure 58: Configuring the A/D Subsystem To Acquire Data in Burst Mode

Note: In general, it is more precise to specify burst duration in terms of number of samples instead of number of seconds. In this way, you know exactly how many samples to expect.

Using an External Trigger

To make things a little more interesting, we will add an external trigger to the program. Suppose the hammer mechanism outputs a steady TTL signal when it is ready to drop the hammer. We will use this TTL signal as the trigger to start acquiring the burst of data. Following the instructions in your hardware manual, connect the signal from the hammer mechanism to the external trigger input on the screw terminal panel. You also need to connect the ground.

Now, click the **Timing** button in the A/D Advanced Configuration dialog box. Leave the clock source as Internal; we are not changing the clock. Select **External** under Trigger Source, and leave the mode as Normal. Click **OK** to close the Timing dialog box.

Now that you have defined the channel list and set up an external trigger, click **OK** to close the dialog box.

Example on Disk: The TRIGGER.VEE example program (in the \Examples\Dataacq\ directory) shows how to set up a user-selectable trigger. This example does not use burst mode. The Set Trigger object receives input from an Enum object which indicates whether the user wants to use an internal or external trigger. The Set Trigger object overrides any trigger settings in the Timing dialog box.

What's Really Going On?

In this example, the A/D Config object executes first, configuring the A/D subsystem for burst mode and setting up the external trigger. Only enough memory for the data queues is set aside for the amount of data specified in the burst mode section of the A/D Configuration dialog box. When execution passes to the Get Data Panel objects, the subsystem is running, but does not start acquiring data because it is waiting for an event to trigger it. When the hammer starts falling, a TTL signal is output (our external trigger), and the subsystem begins acquiring data. As usual, the data is stored in the Mixed Data Buffer, and then pulled out, sorted by channel, and placed in the data queues. When each data queue has exactly 1024 samples in it, the acquisition stops on that channel. When acquisition stops on all channels, the subsystem stops. The Get Data Panel object retrieves the number of samples specified in its Points edit box, and passes this data to other objects in the program.

Note: Because only enough memory is set aside for the specified number of samples or number of seconds of data in the data queues, using burst mode results in more efficient use of memory.

Suppose you set the Get Data Panel object to retrieve 1025 samples, and enable burst mode for 1024 samples. Because the Get Data Panel object tries to retrieve more data than has been acquired, error 2012: "The acquisition or output is complete" occurs. To correct this error, set the Get Data Panel object to retrieve 1024 samples.

Suppose further that burst mode is set to acquire 1024 samples again, but the program uses a looping construct whereby the Get Data Panel object retrieves 512 samples three times, for a total of 1536 samples. An error is raised in this situation as well, because the Get Data Panel object is trying to retrieve more data than is stored in the data queues. To correct this error, either reduce the loop to two iterations, so the total number of samples retrieved is exactly 1024, or increase the number of samples in the Burst mode configuration to 1536.

Acquiring Pre-Trigger Data

Sometimes you want to acquire data before a trigger event occurs. This type of operation is called *pre-trigger* acquisition. For example, assume you are performing in-vehicle testing and you want to measure data on channel 0 before the car crashes and plot the pre-trigger data on a strip chart. Figure 59 shows how the program might appear.

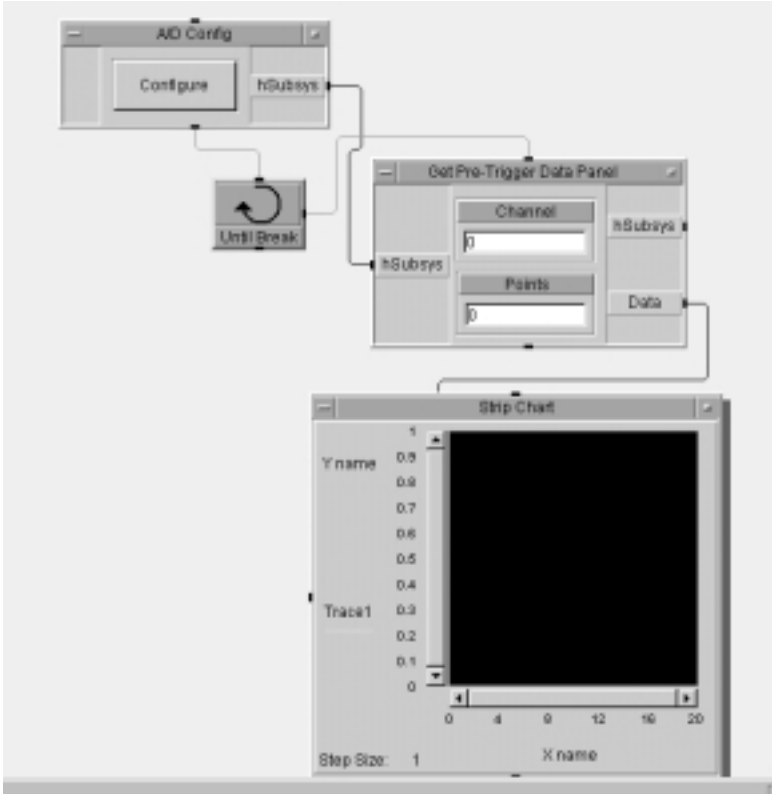

Figure 59: Pre-Trigger Mode Configuration

Configuring for Pre-Trigger Acquisition

To acquire pre-trigger data, first set up the acquisition as you normally would, specifying the channel parameters for your application using the A/D Config dialog box.

Then, click the **Timing** button in the A/D Advanced Configuration dialog box, shown in Figure 60. In the Trigger section (the lower part of the dialog box), select a Pre-trigger Source. Note that the available pre-trigger sources depend on the capabilities of your board; refer to your board and/or device driver documentation for more information.

Specify a post-trigger source in the list box next to the Source; this setting indicates the trigger that stops pre-trigger acquisition and starts the post-trigger acquisition. (Note that when collecting data both before and after a trigger event, the operation is often called *about-trigger* acquisition.)

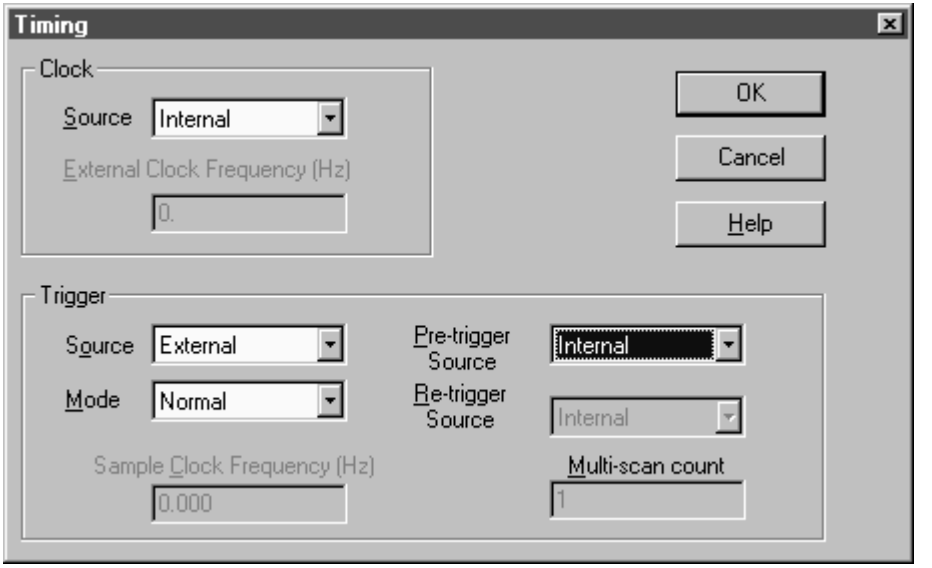

Figure 60: Specify Pre-Trigger and Post-Trigger Sources

Acquiring and Retrieving Pre-Trigger and Post-Trigger Data

When it receives the trigger pulse from the pre-trigger source, the board begins acquiring pre-trigger data. When the trigger pulse from the post-trigger source occurs, the board stops pre-trigger acquisition and starts post-trigger acquisition.

When you specify a pre-trigger source, the file name in the Advanced dialog box for individual or multi-channel file output represents the file where the post-trigger data is stored. A separate file stores the pre-trigger data. The name of the file that stores the pre-trigger data is the same as the post-trigger file name, but contains the letter "P" at the end or replacing the eighth character of the base file name. For example, if your post-trigger file name is *aindata.dtv*, the pre-trigger data file name is *aindatap.dtv*. Likewise, if your post-trigger file name is *chn0data.dtv*, the pre-trigger data file name is *chn0datp.dtv*.

Note: You can specify the post-trigger file name only.

To retrieve pre-trigger data, use the Get PreTrigger Data Panel or Get PreTrigger Data Function object. To retrieve post-trigger data, use the Get Data Panel or Get Data Function object.

Acquiring Data Using a Triggered Scan

Sometimes you want to acquire data on several channels and have each channel sampled whenever a trigger pulse is received. This type of data acquisition is called *triggered scan*.

First, configure the input channels as you normally would, setting the channel's rate to the frequency at which the trigger pulse is expected (in triggered scan mode, this is the re-triggering rate). For example, if the trigger is expected 70 times per second, set the channel's rate to 70 Hz, as shown in Figure 61.

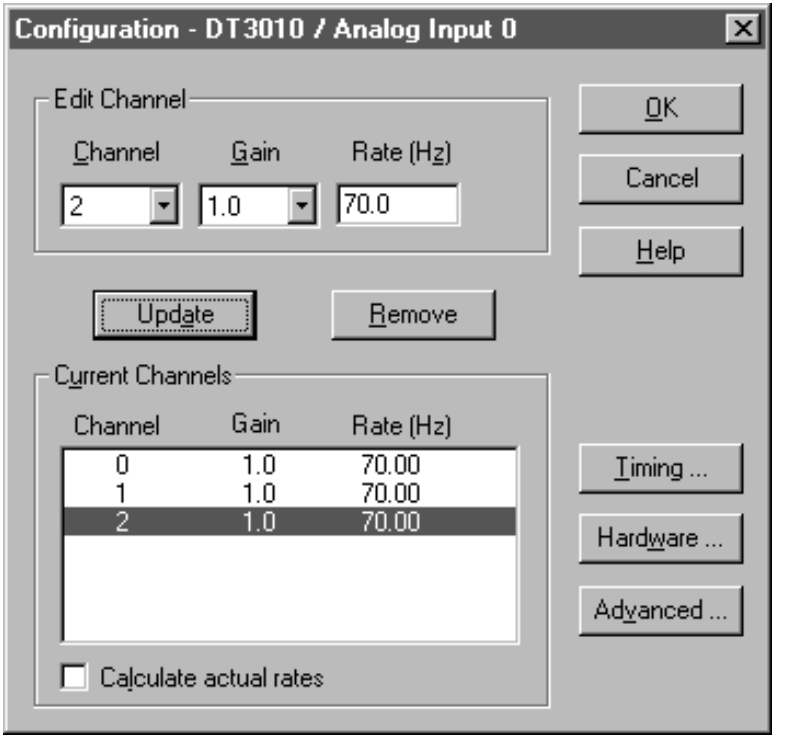

Figure 61: Setting Up the Channels and the Channel Rate for Triggered Scan

To configure a trigger and retrigger, click the **Timing** button in the A/D Advanced Configuration dialog box, shown in Figure 62. In the Trigger section (the lower part of the dialog box), indicate whether the initial trigger is externally supplied (**External**) or internally supplied (**Internal**) using the list box next to Source. Select **Triggered Scan** from the list box next to Mode and select the Re-trigger Source; this trigger source retriggers the scan once it is started.

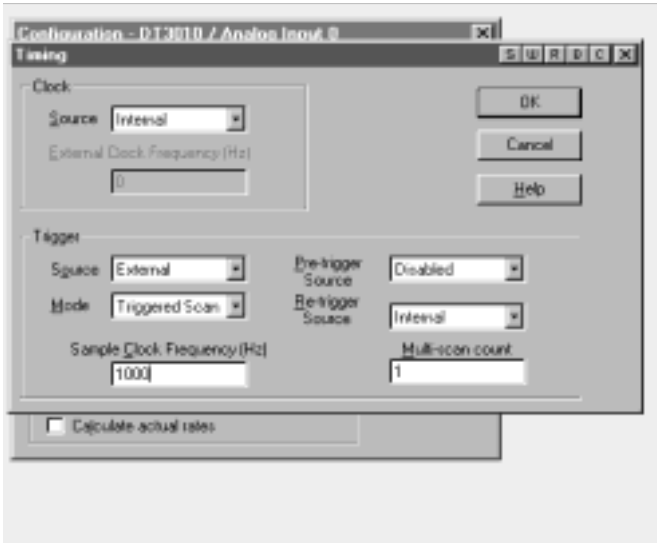

Figure 62: Setting Up Triggering for Triggered Scan

Then, set the Sample Clock Frequency, which is the frequency between channels in the scan. For example, suppose you want to sample two channels 1 ms apart each time a trigger pulse is received. The sample clock rate is 1 channel per 1 ms, or 1000 cycles/1000 ms; this corresponds to a sample clock frequency of 1000 Hz.

If you want to acquire several samples from several channels each time a trigger or retrigger pulse occurs, you can also set the multi-scan count. For example, if you want to acquire eight samples from two channels each time a trigger or retrigger occurs (and perhaps average these samples later), set the multi-scan count to 8. Note that if you specify a multi-scan count, ensure that the channel frequency that is entered in the Rate box of the A/D Config dialog box is the result of the expected trigger frequency multiplied by the multi-scan count. For example, if you expect a trigger every 20 Hz and the multi-scan count is 8, enter a channel rate of 160 Hz.

Assume, for example, that you want to sample three channels (0, 1, and 2) each time an external trigger pulse occurs. The initial trigger is an external trigger; the retrigger is an internal trigger. Trigger pulses are expected 10 times per second (or 10 Hz) and you want the period between each channel in the scan to be 0.1 ms (or 10 kHz). Lastly, you want to acquire seven samples for each channel in the scan. Set up the triggered scan as follows:

- In the A/D Config dialog box,
  - Add channels 0, 1, and 2 to the Current Channels List; and
  - Set the channel rate to 70 Hz (the expected trigger frequency multiplied by the multi-scan count);
- In the A/D Timing dialog box,
  - Set the Trigger Source to External;
  - Set the Trigger Mode to Triggered Scan;
  - Set the Re-trigger Source to Internal;
  - Set the Sample Clock Frequency to 10 kHz; and
  - Set the Multi-scan count to 7.

After two trigger pulses, the order in which the channels were sampled in this example appears as shown in Figure 63.

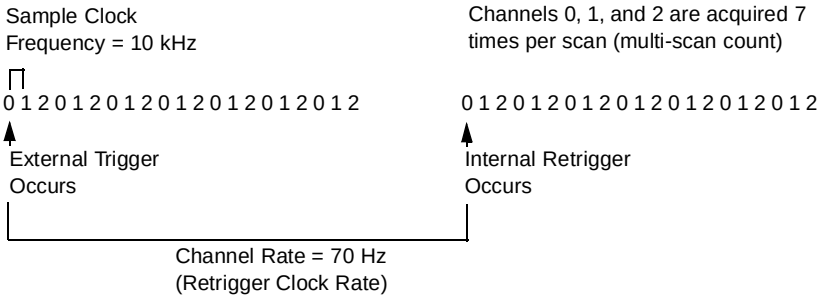
Sample Clock
Frequency = 10 kHz
0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
External Trigger
Occurs
Channels 0, 1, and 2 are acquired 7
times per scan (multi-scan count)
0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
Internal Retrigger
Occurs
Channel Rate = 70 Hz
(Retrigger Clock Rate)

Figure 63: Channel Sampling Order

Example on Disk: The TRIGSCAN.VEE example program (in the \Examples\Dataacq\ directory) shows how to acquire data on several channels by scanning through the channel list repeatedly. Each scan is triggered by a trigger pulse. This example requires a data acquisition board that supports triggered scan with an internal trigger.

Table 4 lists hardware having A/D subsystems that support DT VPI triggered scan mode, triggers, and clocks.

Table 4: Hardware Support for Clock and Trigger Combinations for A/D Subsystems

Board	Clock	Internal Triggers	External Triggers	Internal Pre-Triggers	External Pre-Triggers	Threshold Triggers
DT2821 Series	Internal External		**X** **X**			
DT2831 DT3831 Series	Internal External	**X** **X**	**X** **X**			
DT2839	Internal External	**X** **X**	**X** **X**			**X** **X**
DT300 Series	Internal External	**X** **X**	**X** **X**	**X** **X**		
DT3000 Series	Internal External	**X** **X**	**X** **X**			
DT3010 Series	Internal External	**X** **X**	**X** **X**	**X** **X**		**X** **X**
DT9800 Series	Internal External	**X** **X**	**X** **X**			

Changing the Subsystem Configuration

The **DT DataAcq** menu provides objects to check and change the current subsystem configuration programmatically. In previous examples, you specified the desired configuration using the subsystem configuration object (A/D Config, D/A Config, Digital In Config, or Digital Out Config), before you ran the program.

There are two types of configuration objects:

- Those that obtain (or "get") the values of certain parameters for a given subsystem, and
- Those that change (or "set") the values of certain parameters.

Like the other data acquisition objects, the set and get configuration objects require a subsystem handle as one of the data inputs. Table 5 summarizes the configuration objects and their purpose.

Table 5: Summary of the Get and Set Config Object s

Name	Description
Get and Set Sampling	Obtain or specify the sampling rate and gain settings for each channel in a given subsystem.
Get and Set Clocking	Obtain or specify the clocking source and aggregate clock frequency for a given subsystem.
Get and Set Channel Buffer Size	Obtain or specify the buffer size of a given channel(s) for a given subsystem.
Get and Set Range	Obtain or specify the programmable voltage range for a given subsystem.
Get and Set Trigger	Obtain or specify the trigger sources and/or trigger mode for a given subsystem.

Table 5: Summary of the Get and Set Config Objects (cont.)

Name	Description
Get and Set Sample Clock Frequency	Obtain or specify the channel-to-channel sample frequency for an internal re-trigger and triggered scan of channels.
Get and Set Channel Filter	Obtain or specify the lowpass (cutoff) frequency for a given channel on a given subsystem.
Get and Set Data Mode	Obtain or specify the data I/O management configuration for the given channel(s) for a given subsystem.
Get and Set Multi-Channel Mode	Obtain or specify the use of multi-channel file output, aggregate rate, and automatic build of the channel-gain list.
Get and Set C/T Mode	Obtain or specify the mode, gate source, gate type, and pulse type of a C/T subsystem.
Get and Set C/T Frequency	Obtain or specify the frequency and duty cycle of a C/T subsystem.
Get and Set C/T Pulse Width	Obtain or specify the delay and pulse width of a C/T subsystem.
Get and Set C/T Measure Duration	Obtain or specify the duration over which to measure events.

For more information on the get and set config objects, refer to DT VPI online help.

Often, the get and set config objects are used as a pair. Place the get config object so that it executes before the set config object. Then connect the outputs from the get config object to the inputs of the set config object, except for the one or two parameters you intend to change.

CAUTION:

Because the set config objects change the configuration of the subsystem, it is important that you not reconfigure a running subsystem.To avoid this situation, stop a running subsystem with a Stop Acq object before the set config object executes.

For example, the program shown in Figure 64 uses the Get Sampling and Set Sampling objects to change the analog input rate (the sampling rate). The A/D Config object is configured to output a 1000 Hz sampling rate on channel 0.

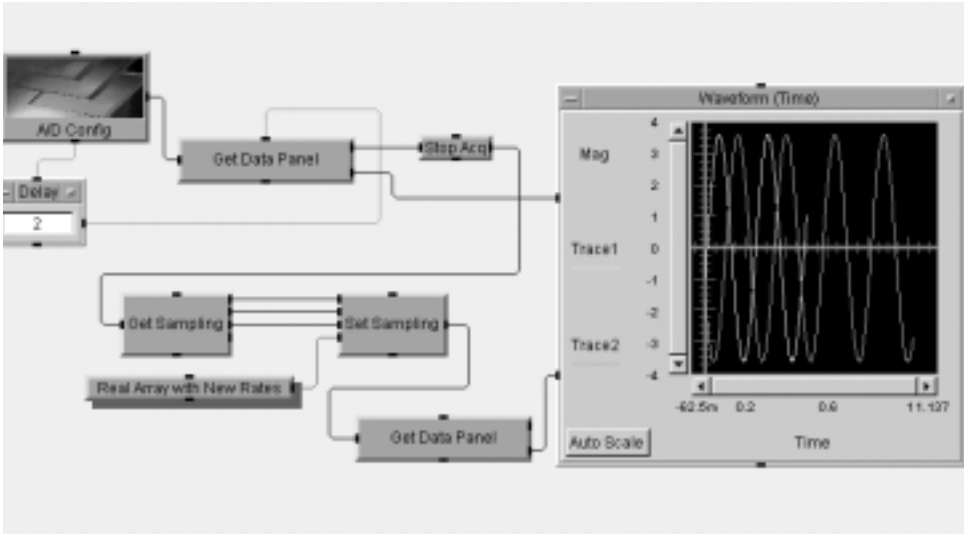

Figure 64: A ProgramTo Change the Output Rate Programmatically

When you run this program, 500 points (one-half second at a 1000 Hz sampling rate) is acquired by the A/D subsystem and then is retrieved by the Get Data Panel object. Since that is all the data we need for this example, we stop the subsystem. Otherwise, it would

continue to acquire up to 4 seconds worth of data.

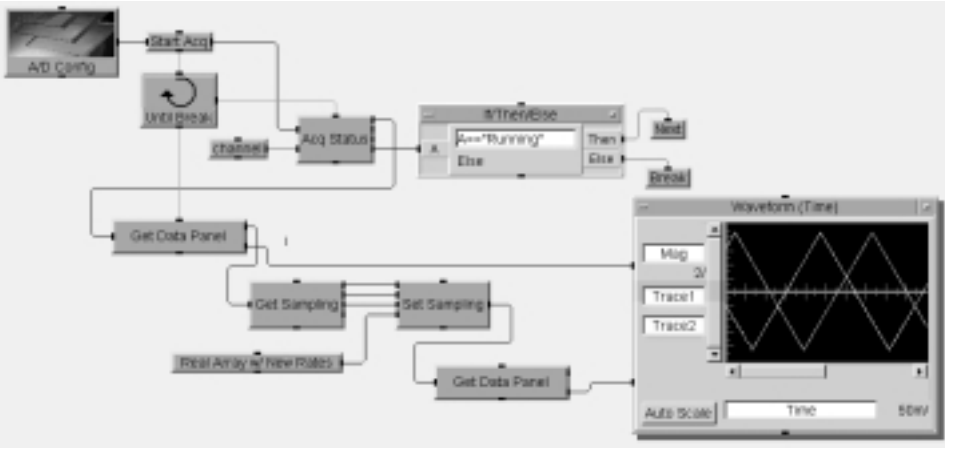

Figure 65: The Same Program Using Acq Status Instead of Delay

The Get Sampling and Set Sampling objects, used as a pair, allow a change to the A/D sampling rate. Because the Set Sampling object expects the Rate input as a real array containing one value for each channel, the new rate is in an real constant object. We set this object's property for a one-dimensional array of size 1 (we are only using one channel), which is the sampling rate in hertz.

When the second Get Data Panel object receives the same data and a handle to a reconfigured A/D subsystem, it acquires the data using the new sampling rate.

Handling Acquisition Errors

In general, if an error is raised during acquisition, a dialog box appears and your program stops running. To clear the dialog box, click **OK**. Any data in memory at the time is flushed out. You need only click **Run** Start to restart your program. For example, if a Get Data Panel or Get Data Function object tries to retrieve data for a channel and not enough samples are available, an error is raised (see Figure 66). Similarly, if the Put Data Panel or Put Data Function object is executed before it has at least 0.5 seconds of data, an error is raised. In both cases, the subsystem is no longer running, so a Stop Acq object is not needed.

If a Get Data Panel or Get Data Function object tries to retrieve data for a channel that has stopped, an error is raised (see Figure 66). Similarly, if the Put Data Panel or Put Data Function object is executed when it has less than 0.5 seconds of data, an error is raised. In both cases, the subsystem is no longer running, so a Stop Acq object is not needed. Any data in memory at the time is flushed out. You need only click **Run** or **Start** to restart your program.

Error Message
The Acquisition/Output is complete for the channel. This is typically caused by processing that can not keep up with the acquisition or output rate.
in Compiled Function 'GetData'
Object Title: GetDataPrimitive
Object Type: Call
Error number: 2012
OK

Figure 66: Error Is Raised when Not Enough Data Is Available in Memory

Errors When Writing to File

When writing data directly to a file, you want to be sure of the data's integrity. Therefore, you want to know immediately of any error that occurs—whether it is an output error, a read error, a problem with the file, and so on.

There are two ways to monitor file errors. One way is to enable PopUp File Error detection in the Read from File edit box. Enabling this feature permits errors to appear immediately in a pop-up dialog box. This error cannot be trapped. The other way is to use the Acq Status object to repeatedly poll the subsystem to determine if file output stopped for any reason.

Trapping Errors

If you wish to trap the errors for other purposes, such as logging them for a report, you need to add an Error output pin to the data acquisition object. However, because most data acquisition objects are secured UserObjects, you cannot directly add an error terminal as you can to other DT VPI objects. To verify this fact, try to select **Add Terminal** from Get Data Panel's object menu. Notice it is grayed out.

To add an error output pin to a data acquisition object, select the object and put it into a UserObject (**Edit ⇨ Create UserObject**). Then, select the User-Object's object menu ⇨ **Add Terminal** ⇨ **Error Output**. The error output pin is usually connected to the sequence in pin of a Stop Acq or Close Subsystem object, if present.

For example, take a look at the program shown in Figure 67. The A / D subsystem is configured to acquire 512 samples in burst mode, but the Get Data Panel object has 10000 points in its Points edit box. When you run this program, 512 samples are acquired. These values are stored in memory, and when the program executes, Get Data Panel tries to retrieve 10,000 values! Obviously, there is not enough data stored, so an error is raised. The subsystem stops running. The Error output pin fires, activating the Close Subsystem object.

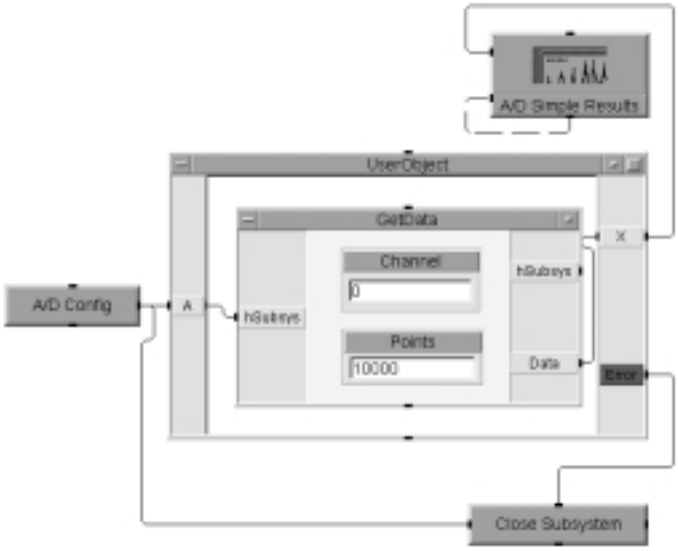

Figure 67: A ProgramToTrap an Error

The Close Subsystem object requires the subsystem handle as input. In the example, the Close Subsystem object gets the subsystem handle from the A/D Config object. Recall that the Get Data Panel object also outputs the subsystem handle. However, in this example, the subsystem handle from Get Data Panel is not sent to the Close Subsystem object because if an error is raised in the Get Data Panel object, all of its output data is invalid.

Note: If an error is raised with a data acquisition object, the hSubsys output on that object does not have valid data, and therefore does not contain a valid subsystem handle.

The Stop Acq and Close Subsystem objects are control objects. The Stop Acq object is used to halt data acquisition on the hardware and flush any unread data from the memory buffer. The Close Subsystem object terminates a subsystem's configuration and deletes the subsystem handle. A currently running subsystem stops. Both objects require the subsystem handle, as do the other data acquisition objects.

Example on Disk: The TOFILE.VEE example program (in the \Examples\Dataacq\ directory demonstrates how to monitor the progress of file output from the A/D subsystem and handles acquisition errors.

Sending Data to a Disk File

There are two methods to send acquired data to disk. One way is to use the VEE To File object (**I/O ⇨ To ⇨ File**). In this case the data is written after converting it to voltage values, and the program can run only as fast as the To File object can run. The other way is to send the data directly to disk immediately after acquisition, circumventing having VEE write to the file. In this case, your program can sample at much higher sampling rates with gap-free data. These two methods are described in the following subsections.

Writing to File Using the To File Object

Writing data to disk with the To File object is the easiest method because you can read the data back using the From File object without having to handle conversions or read a header. However, to use the To File object, your sampling rate requirements must be under 15 kHz when writing binary files or under 8 kHz when writing text files. The actual sampling rate threshold is system- and board-dependent.

The example shown in Figure 68 shows how to write acquired data to a text file using a To File object for each channel.

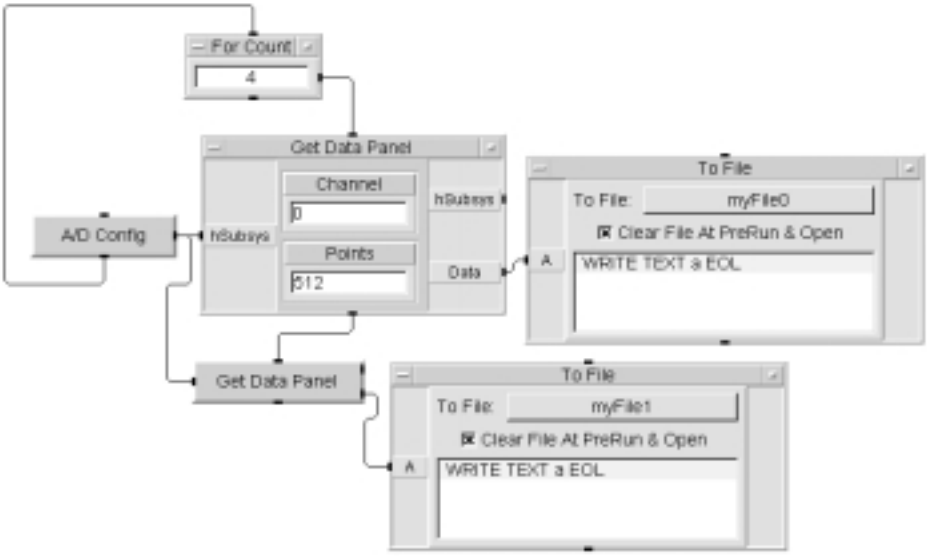

Figure 68: Using the To File Object To Write Data to a Text File

The A/D Config object configures the A / D subsystem for two channels, each sampled at 1 kHz. Burst duration is enabled for 2048 samples. Because the Get Data Panel objects are set to retrieve 512 samples, you have to loop four times to read all the collected data samples (512 * 4 = 2048 samples). Figure 69 shows the A / D Advanced Configuration dialog box.

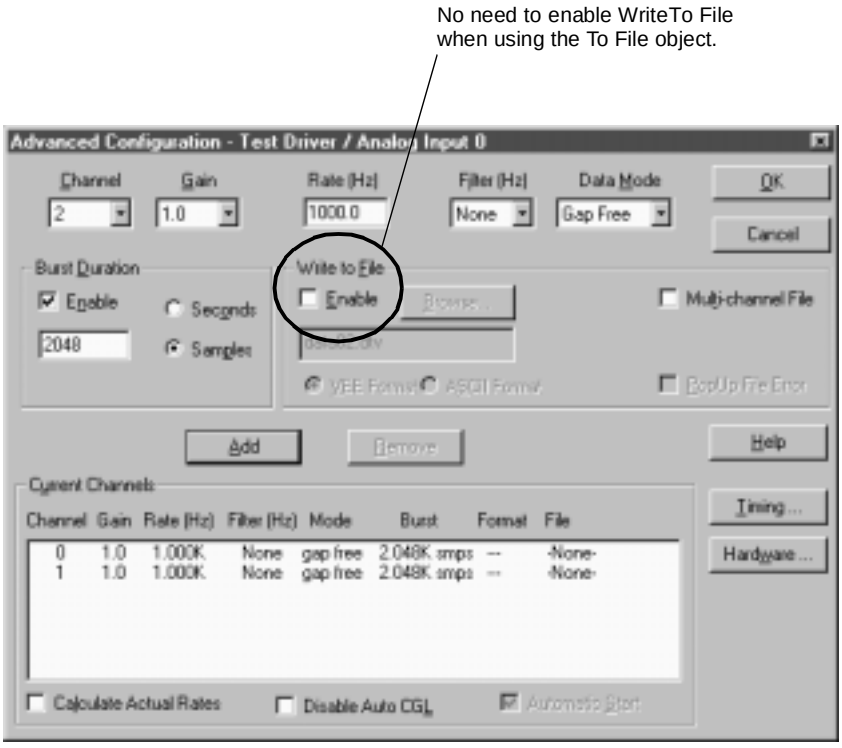

Figure 69: Configuring To Write to File Using the To File Object

When you run this program, the A/D subsystem acquires a burst of 2048 samples and stores them in the Mixed Data Buffer. The two Get Data Panel objects each retrieve 512 samples from the individual channel's data queues every time the For Count object loops. Two files are written, MYFILE0 and MYFILE1. In this example, they are text files.

Figure 70 shows what MYFILE0 looks like in Windows WordPad.

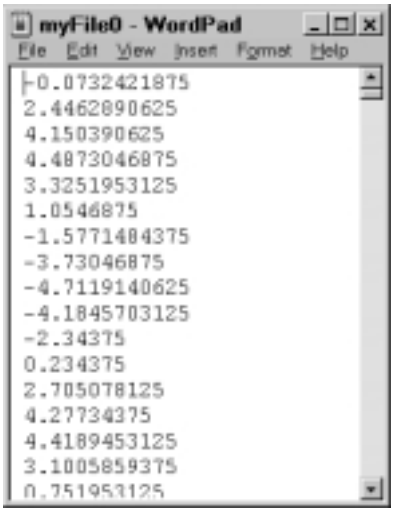

Figure 70: Looking at a Text File of Acquired Data

It is easy to read this data back into VEE. The example shown in Figure 71 shows a From File object reading in MYFILE0, which we know contains 2048 points. The data is displayed in a Logging AlphaNumeric object and in an XY Trace object that has been autoscaled.

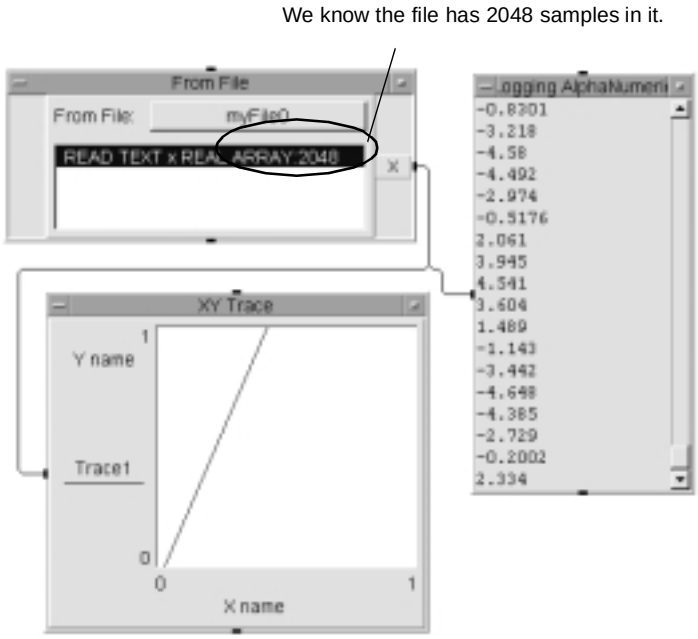

Figure 71: A Program To Read Acquired Data from a Disk File

Another way to use the To File object even if your sampling rate requirements are not in the 15 kHz or less range is to acquire the data first, and *then* write it to a file. The example shown in Figure 72 uses the Acq Status object to determine when acquisition is done. When acquisition is done, the Get Data Panel object is executed and retrieves 9865 points from the channel's data queue. This data is then written to a file (MYFILE) in binary format. The data in this file can also be read using the From File object.

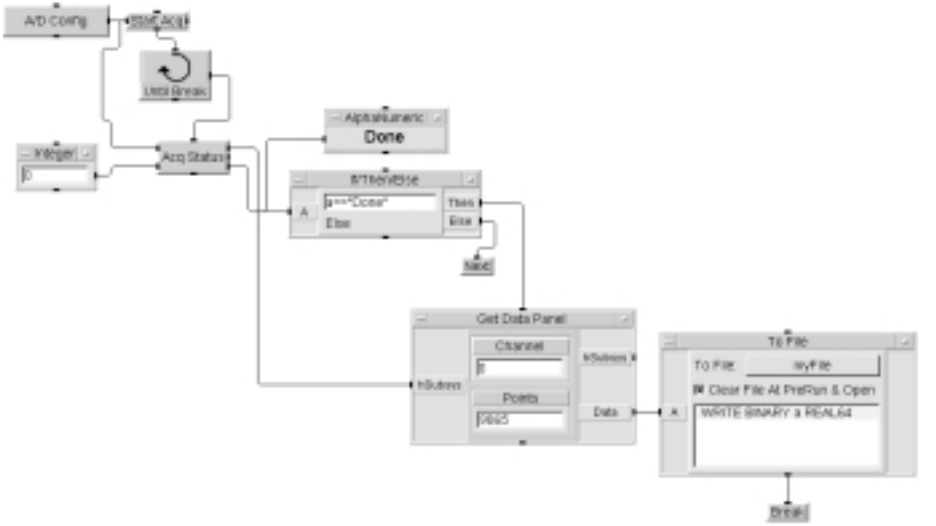

Figure 72: Writing Data to a File after It Is Acquired

Because the data is written to disk after it is acquired, or *post-processed*, you can run this example at a high sampling rate, such as 200 kHz. Note that you should enable burst mode for 9865 samples in the A/D Advanced Configuration dialog box.

Writing Directly to File

If you need to acquire and store data very quickly (over 8 or 10 kHz), DT VPI provides the capability to pass the data directly to a disk file immediately after it is acquired, even before the Get Data Panel object has a chance to retrieve it! Data handled this way is stored in raw, unprocessed format to save the time needed to convert from raw binary coded values to voltages. The binary file contains a special file header, which includes all the information necessary to convert the raw data at a later time, including the coding method used (offset binary or twos complement), the resolution, and the voltage range.

This file cannot be read into VEE using the From File object, unless the necessary steps are taken to convert the raw data.

You can write data for each channel to a file, or you can write data for several channels to a single file. The file header for VEE format and VEE multi-channel format are described in Appendix C.

CAUTION:

It is important to use burst mode when acquiring and writing data directly to disk. Since clicking the Stop button on the tool bar does not stop data acquisition, the subsystem continues to run and write data to the disk file until the data queues and the Mixed Data Buffer are filled. If the system keeps up with the data and the Mixed Data Buffer never fills, you could fill your hard disk with data. The only way to stop data acquisition is to reconfigure the subsystem or execute a Stop Acq object.

However, by using burst mode, you specify the exact amount of data you want and thus the size of the file. A warning is raised (if you have also chosen Allow Gaps data mode) that your disk may become filled. If you do not plan on using burst mode when writing data to disk, be sure to use a Stop Acq object in your program.

Example on Disk: The DTVFILE.VEE example program (in or \Examples\Dataacq\ directory) illustrates how to read a file containing raw, unprocessed data and convert it using VEE objects and the information found in the file header.

The TOFILE.VEE example program (in the \Examples\Dataacq\ directory) demonstrates writing data to disk and monitoring for errors.

You may want to enable PopUp File Error detection in the Write to File edit box. Enabling this feature permits errors to appear immediately without having to use the Acq Status object to repeatedly poll the subsystem to determine if file output stopped.

In the example shown in Figure 73, we will look at writing data directly to disk files, without using the To File object. We will set it up to acquire data on channels 0, 1, and 2, write the data from channels 0 and 2 to disk files, and display only the data from channel 0. Use the program we created in the section "Acquiring Analog Data Continuously," starting on page 96.

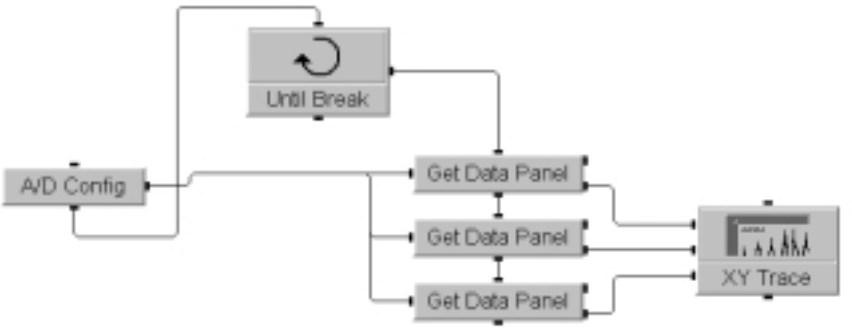

Figure 73: A Program To Write Acquired Data to Disk Files

Configuring Each Channel for Writing to a File

Some applications require the greater precision of data saved in a separate file for each channel. Output may be in either VEE (binary) or ASCII format. You can choose one using the **VEE** or **ASCII Format** buttons in the Write to File section of the A/D Advanced Configuration dialog box. If you select VEE format, data for each channel is written to a separate file in raw, unprocessed format. This format is not readable in a word processor, but includes a file header with information for DT VPI to "decode" the binary file. If you select ASCII format, data for each channel is written to the file as voltage values. ASCII format should only be used with slower sampling rates due to the time needed to convert the data to ASCII voltage values while acquiring data.

Note: To write the data for several channels to a single file, place an X in the checkbox next to Multi-channel File and refer to "Configuring Multiple Channels for Writing to a Single File," starting on page 135 for information.

Using the A/D Advanced Configuration dialog box, you can set up the channel list just as you would normally. As you add each channel, click the checkbox labeled Enable in the Write to File section. In this example, the VEE format is selected.

Set the channels to the following rates and gains:

Channel	Rate (Hz)	Gain	Write to File Enabled?	File Name	Burst Enabled ?	Burst Duration (seconds)
0	10000	1	yes (VEE format)	DATA00.DTV	yes	5
1	10000	1	no	n/a	yes	5
2	10000	1	yes (VEE format)	DATA02.DTV	yes	5

Figure 74 shows how the dialog box should look when you are done.

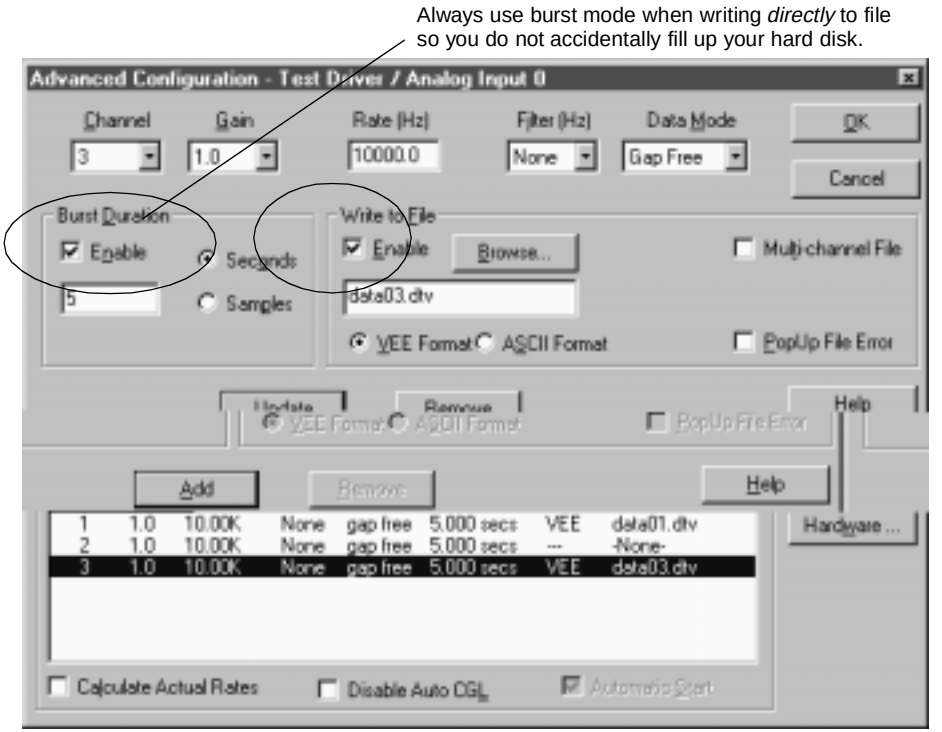

Figure 74: Configuring the A/D Subsystem To Write Data to Single-Channel Files

When this program is run, the A/D subsystem is configured, and data from channels 0, 1, and 2 is acquired and stored in the Mixed Data Buffer. This data, as usual, is pulled out and sorted into the data queues, which are sized by the burst duration specification. At the same time, data is written directly to DATA00.DTV and DATA02.DTV, respectively. Each of these files contains raw, unprocessed values and a special file header. The file header includes information for the D/A subsystem to convert the values into voltages at a later time.

Configuring Multiple Channels for Writing to a Single File

You can opt to save data from all specified channels to a single binary data file. This option is appropriate for high channel count applications, situations where you want to sample at high sampling rates, or you are near the maximum number of open files permitted by your system. Choose this mode by placing an X in the checkbox next to Multi-channel File in theWrite to File section of the A/D Advanced Configuration dialog box. The data is always gap-free. Each channel's sampling rate is the same; the rate is computed by dividing the aggregate rate by the number of channels. Your choice of aggregate rate is determined by the data acquisition subsystem capabilities. This data format uses the VEE Multi-channel File Format (see Appendix C for file header information).

In the example shown in Figure 75, three analog input channels are used (0, 1, and 2). There is a Get Data Panel object for each channel, and the number of points is 512. Since the For Count object loops twice, the total number of points retrieved is 1024. The first channel is displayed on the XY Trace object.

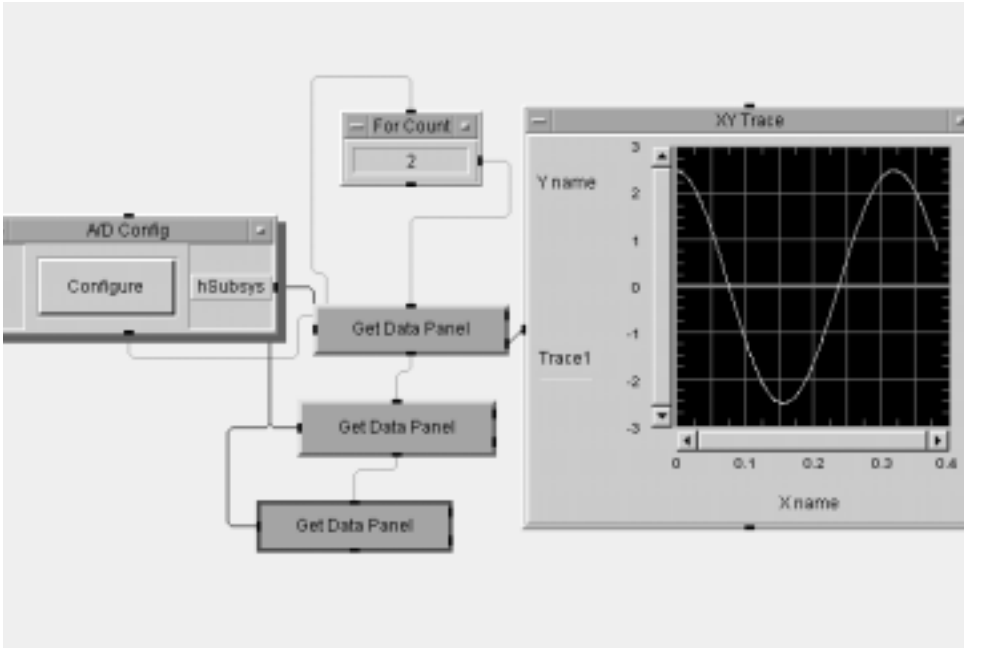

Figure 75: An Example Program for Writing Three Channels to One File

To configure for this example, click the **Configure** button on the A/D Config object. Click the **Advanced** button. To avoid accidentally filling up your hard disk, enable burst mode by placing an X in the checkbox next to **Enable** in the Burst Duration section. Make sure 1024 samples is selected for the burst duration (512 points * 2 loops). Enable the **Multi-channel File** option by clicking the checkbox. Leave the default file name, MULTCHAN.DTV. The format is always VEE (binary) when using the Multi-channel File option.

Enter an aggregate sampling rate, such as 4000 Hz, in the **Aggregate Rate (Hz)** edit box. Then click the **Add** button three times to add three channels. Figure 76 shows how the dialog box should look when you are done configuring.

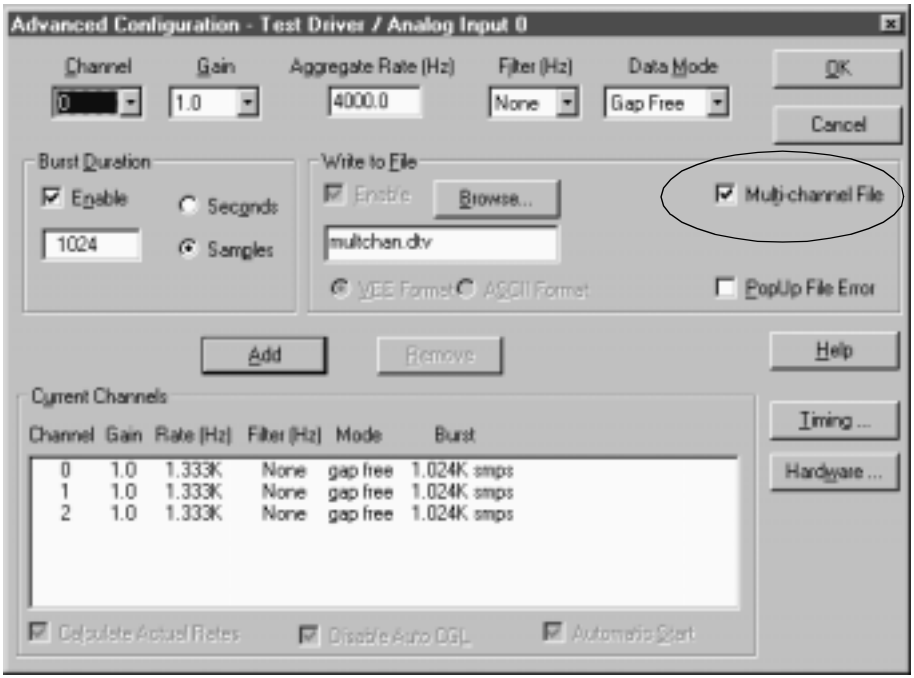

Figure 76: Configuring the A/D SubsystemTo Write Three Channels to a Single File

Click **OK** to close the dialog box. To run the program, click the **Run** button. For each of the three channels, 1024 samples are acquired. This data is retrieved by the Get Data Panel objects in two passes of 512 points each. Most importantly, this data is written to the file MULTCHAN.DTV in raw, binary format while the data is being acquired.

Example on Disk: The MULTFILE.VEE example program reads in a multiple channel file named MULTCHAN.DTV. It displays one channel from the file at a time, offering a little dialog box for you to specify which channel you are interested in.

The ALLCHAN.VEE example program illustrates how to create a multiple channel file without using the A/D Advanced Configuration dialog box. Instead, the program uses the Get-Multi Channel Mode and Set Multi-Channel Mode objects as well as the dtv.GetFileSize advanced function. The file it creates (MUTLCHAN.DTV) can be read by the MUTLFILE.VEE program.

Configuring for Slow Sampling Rates

By default, the Disable Auto CGL option in the A/D Advanced Configuration dialog box is unchecked (enabled). DT VPI automatically builds the channel-gain list so that the specified channels and their specified rates can be achieved. To achieve the sampling rate closest to that specified for each channel, data may actually be collected for unspecified channels but never used.

However, the automatic channel-gain list is somewhat inefficient when using low sampling rates (less than 1 kHz). Therefore, you may want to avoid having DT VPI automatically build the channel-gain list. To do so, place an X in the checkbox next to Disable Auto CGL. This disables the automatic build of the Channel-Gain List and causes data to be sampled only from the specified channels. Individual channel sampling rates are ignored; instead the aggregate rate is set and the channel rates are equal (aggregate rate divided by the number of channels).

Sending Data Through DDE

Dynamic Data Exchange, or *DDE*, defines a message-based protocol for communication between Windows applications. The communication takes place between a *DDE client* and *DDE server*. The DDE client requests a conversation with the DDE server about a certain topic. The client can then request information or services from the server, send information to the server, or wait. The server responds by sending data or executing procedures.

Note: See "Using Dynamic Data Exchange (DDE)" in the *VEE Advanced Techniques* or *VEE OneLab Advanced Techniques* manual.

Windows applications that support DDE can act as either a client or server. As a client, VEE programs can initiate a conversation with an application that supports DDE and send or request data from that server.

The example shown in Figure 77 uses Microsoft Excel as the server application. A spreadsheet application is appropriate because it can be used to manipulate the acquired analog data that our program sends to it. You need to have Excel installed and in your DOS path, already running, or launched by an Execute Program (PC) object.

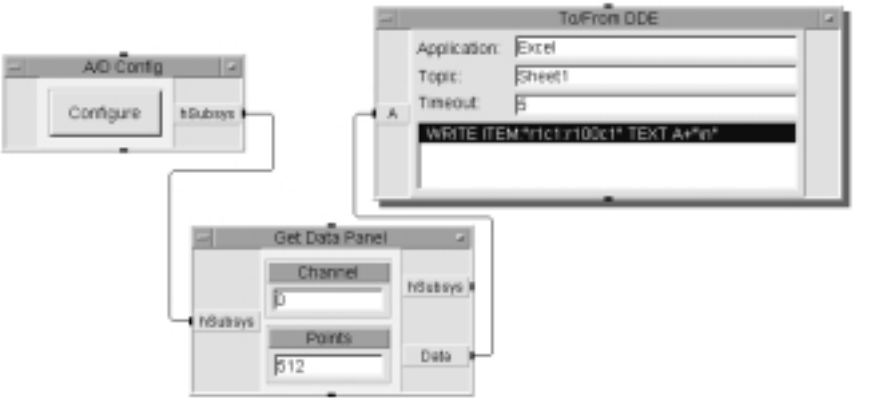

Figure 77: A Program To Send Acquired Data to Excel through DDE

In the To/From DDE object (**I/O ⇨ To/From DDE (PC)**), by default the Application is Excel and the Topic is Sheet1, an Excel document. To fill in the information for the transaction, double-click the empty horizontal bar below Sheet1 and fill in the dialog box as shown in Figure 78.

Figure 78: DDE Transaction Settings To Send Acquired Data to Excel

Click READ (REQUEST) and change it to **WRITE (POKE)**. Click STRING FORMAT and change it to **DEFAULT FORMAT**.

When working with DDE and using VEE as the client, you typically need to find out how the server application expects the item in the conversation to be phrased. The Item is a string specifying the starting row and column and the ending row and column to which the data should flow. A colon separates the starting and ending points. For example, if you have 100 values in a one-dimensional array, you might specify the Item as "r1c1:r100c1", which says to put the data in one column, from row 1 to row 100. The text"\n" instructs Excel to go to the next row. In the TEXT edit box, enter the name of the one-dimensional array containing the data. (You also need to add an input terminal for the data.)

When you run this program, 512 samples of data are acquired on channel 0. A conversation with Excel is initiated by the To/From DDE object. If VEE can find the server application, VEE launches it. Then the data is sent to Excel, filling row 1, column 1 to row 100, column 1. To verify that the data was sent, open the file Sheet1.

Example on Disk: If you want to programmatically control the row number, then use string concatenation to include the value of the row in the string for the Item. In the DDE.VEE example program (in the \Examples\Dataacq\ directory), we want to write the data down one column, so we use a For Count object to input the row number and increment it by one each time. In the edit box next to Item, type "r" + (b+1) + "c1", where *b* is the input value for the row number. Because the For Count object starts with 0 and Excel's rows start at 1, we need to add 1 to the row number input.

In the edit box next to TEXT, type a[b], where *a* is the array containing the data you want to write, and *b*, as you know, is the row number. Lastly, you need to add two input terminals to the To/From DDE object.

Optimizing Analog Input

You can optimize analog input with DT VPI by doing the following:

- Use burst mode whenever appropriate.
- When acquiring data continuously, use the Allow Gaps data mode when it is unnecessary to process every point. Allow Gaps is found in the A/D Advanced Configuration dialog box. See the DT VPI online help for more information.
- When using a relatively slow sampling rate, use the Acq Status object to check on how many samples are available before reading them with the Get Data Panel object or Get Data Function object. The Get Data Panel and Get Data Function objects do not return until all of the data points specified have been retrieved from memory.
- When configuring, leave Calculate Actual Rates unchecked until you are done creating the channel list. Doing so avoids having DT VPI take the time to calculate between each channel entry.
- Use the Close Subsystem object after your data has been acquired to release memory.
- Use the Get Data Function object instead of the Get Data Panel object. The Get Data Function object runs more efficiently because it is a function call, not a UserObject.
- Use the Write to File feature if you need to acquire and store data fast, over 10 kHz. Writing data directly to disk uses a binary format for the unprocessed data. To read the data, use DT VPI example program DTVFILE.VEE, which uses the information in the file header to process the raw data values to voltages. When using 16 input channels or more, use the multi-channel feature to write data for all the channels to a single file.

- If you are collecting data quickly, increase the number of points each Get Data Panel object (or Get Data Function object) should retrieve from memory. A seconds' worth of data is typically an appropriate amount.
- Do not type 0 in the Points edit box of a Get Data Panel object. If you enter 0, then DT VPI retrieves all currently acquired data from memory. If you are acquiring data quickly, then, depending on how much memory you have in your system, the amount of data may be too much for DT VPI to handle. Generally, you should not retrieve more than 500,000 points at a time.

Analog Output

Connecting External Devices 147
Configuring for Analog Output 147
Starting an Analog Output 155
Outputting Analog Data from a Single Channel 156
Outputting Analog Data Continuously 159
Outputting a Single Analog Value 164
Outputting a Burst of Data 166
Changing the Subsystem Configuration 170
Handling Acquisition Errors 175
Reading Data From a Disk File 177
Receiving Data From DDE 180
Optimizing Analog Output 183

A process complementary to analog-to-digital conversion, called *digital-to-analog conversion* (D/A conversion), changes digital data into analog voltages. This process permits the computer to drive chart recorders, audio amplifiers, process actuators, and other devices requiring analog input. Many data acquisition boards have both A/D and D/A converters. Typically, data acquisition boards have one or two D/A converters (DACs). Each DAC is a channel.

This chapter discusses how to control your data acquisition hardware to perform analog output operations. It includes information on configuring the DACs and setting the sampling rate. Special attention is paid to setting the sampling rate to output data generated by the Function Generator object.

This chapter also includes many example programs and explanations of using the data acquisition objects provided by D TVPI.You will learn how to create programs to

- Output analog data;
- Output continuously;
- Output a single value;
- Output a known number of samples or output for a known duration;
- Change the configuration while the program is running;
- Handle errors; and
- Read data from a file or from another application through DDE.

Connecting External Devices

Before you can output analog data, connect your device requiring analog input. Your hardware manual has detailed instructions on connecting external devices.

Configuring for Analog Output

Now it is time to start building your data acquisition program. For an analog output program, you need the D/A Config object. Select **DT DataAcq ⇨ D/A Config**, and click **Configure**.

Table 6 directs you to the appropriate dialog box to perform various configurations. These configurations are discussed in the following pages.

Table 6: Configuring the D/A Subsystem

To Configure...	In the D/A Config Object, click...
The D/A channel list	**Configure**
Sampling (output) rates for channels	**Configure**
Sampling rates, burst mode, read from file, and data mode (Gap Free or Allow Gaps) for channels	**Configure**, and then **Advanced**
The output voltage range	**Configure**, and then **Hardware**
The output channel type	**Configure**, and then **Hardware**
The clock source	**Configure**, and then **Timing**
The trigger source	**Configure**, and then **Timing**

Selecting a Board when More than One Data Acquisition Board Is Installed

If you have more than one data acquisition board installed in your computer, a dialog box asks you to specify which board you want to use, as shown in Figure 79.

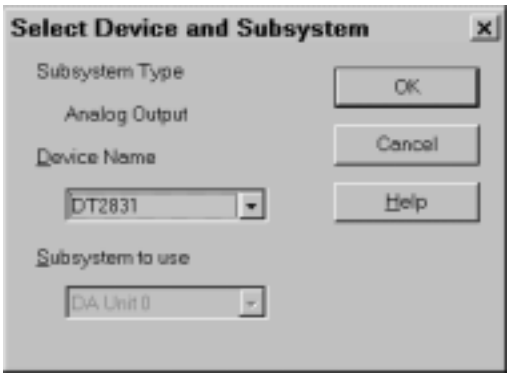

Figure 79: The Select Device and Subsystem Dialog Box

Note: This dialog box appears only if you have more than one data acquisition board installed in your system, or more than one subsystem of that type exists on the board.

Select the name of the board you want to use from the Device Name drop-down list, then select **DA Unit 0** from the Subsystem To Use drop-down list. Click **OK**.

Note: The name of the board is the name assigned to it when its device driver was installed. You should select the board that has the capabilities you need for analog output.

Configuring the Channel List

The D/A Configuration dialog box, shown in Figure 80, is used to build a *channel list*, which specifies which channels you intend to use for analog output, and at what sampling rate and gain.

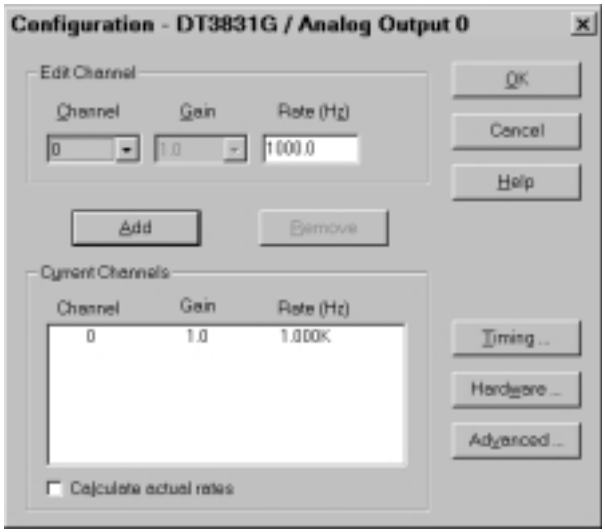

Figure 80: The D/A Configuration Dialog Box

Take a moment to consider the external connections you have made. Most boards have two DACs (channel 0 and channel 1). For each channel you intend to use, enter the gain in the Gain edit box, and the sampling rate needed in the Rate (Hz) edit box, then click **Add**. The channel number, gain, and associated rate are added to the channel list in the Current Channels box. If you need to edit one of the channels, click on the entry in the Current Channels box; the information for that channel appears in the Edit Channels box.

The Calculate Actual Rates checkbox refers to the actual rates that the analog output subsystem can achieve. When this box is checked, the Current Channels box reflects the actual sampling rates. To view the sampling rate that you originally entered, select the channel in the Current Channels box; the original rate for the channel is displayed in the Edit Channel box.

By default, the D/A Configuration dialog box shows channel 0 in the channel list. You can add as many channels to the channel list as your board can accommodate, and you can remove all but one channel. Your channel list is always sorted by channel number, starting with the lowest number. You should note that when you run the program, DT VPI outputs data on the channels in the order that they appear in the channel list.

Selecting an Output Rate

An output rate is required for analog output (except for single value output, which is not clocked). The output rate is used to reconstruct an analog signal from the data values. If you think of the data as the y-values on an X- and Y-axes graph, you need the x-values. The output rate, in terms of samples per second, specifies how far apart to graph the y-values along an x-axis of time.

If you are outputting data that you originally acquired, use the same sampling rate for both analog input and output.

If you are outputting data generated internally in VEE (such as with the Function Generator object), then the process is a little more complicated. For example, suppose you want to output 10 cycles of a cosine waveform on the DAC. Set up the Function Generator object's settings to generate a cosine wave with a Frequency of 1000 Hz and a Time Span of 10 ms. A frequency of 1000 Hz (1000 samples per second) means that each cycle is 1 ms long. If you set the Function Generator object's Time Span for 10 ms, then data for 10 cycles is generated.

You could also set the Function Generator object's Frequency for 2000 Hz and its Time Span for 20 ms and still get 10 cycles.

The number of points generated by the Function Generator object is related to the Rate you set for the DAC in the D/A Config object's configuration dialog box. The Rate setting is actually the sampling frequency, which is the frequency at which the generated data points are output (that is, a high sampling frequency means they are "close together" in time, and a low sampling frequency means that data points are output "far apart" in time). You can adjust the D/A Rate and the Function Generator object's NumPoints setting to achieve the right duration for the output waveform and the right frequency of the output waveform.

Using the example above (frequency of 1000 Hz over a time span of 10 ms), try a D/A Rate setting of 20,000 Hz. Assume you want about 100 points per cycle from the Function Generator; so, for 10 cycles, a setting of 1000 points is needed for NumPoints. To compute the duration of the output waveform, divide the setting for NumPoints by the D/A Rate. For our example, this is

$$1000 points \bullet \frac{1 second}{20000 samples} = duration$$

$$\frac{1}{20} second = duration$$

$$\frac{1000 points}{\frac{20000 samples}{1 second}} = duration$$

Given these settings for the Function Generator object (Frequency of 1000 Hz, TimeSpan of 10 ms, 1000 NumPoints), and with the DAC outputting at a sampling frequency of 20000 Hz, the duration of the 10-cycle output waveform is 0.5 seconds (1/20 second). Because this duration is less than 0.5 seconds, the DAC should be configured for burst mode. In the D/A Advanced Configuration dialog box, enable Burst duration. (See "What's Really Going On?," starting on page 160 to learn about the 0.5 second threshold.)

To compute the frequency of the output waveform itself (in cycles per second, or Hz), divide the D/A Rate (samples per second) by the number of points per cycle, which in our example is 100 points per each of 10 cycles:

$$\frac{\frac{20000 samples}{1 second}}{\frac{100 points}{1 cycle}} = \frac{20000 samples}{1 second} \bullet \frac{1 cycle}{100 points} = 200 Hz$$

The frequency of the output wave is 200 Hz.

If you lower the D/A Rate (remember, this rate is the sampling frequency of the data points), then the points are "spread farther apart," so the duration of the output waveform increases. To achieve a duration long enough to sustain continuous output from the DAC (as opposed to enabling Burst mode), then you should either reduce the D/A Rate to 2000 Hz (1000 points divided by 2000 samples per second yields a duration of 1/2 second), or increase the NumPoints setting in the Function Generator object to 10000 points (10000 points divided by 20000 samples per second also yields 1/2 second).

Example on Disk: The DACONT.VEE example program (in the \Examples\Dataacq\ directory) has a Function Generator object with settings of Frequency of 10 Hz, over a TimeSpan of 1 s, and 1000 points. These settings generate a cosine wave with 10 cycles per second, and 100 points per cycle. The D/A Rate is set for 1000 Hz. The frequency of the output waveform is 10 Hz (1000 divided by 100), over a duration of 1 s (1000 Hz/1000 point).

Selecting an Output Range and Type

To select the output range, click the **Hardware** button from the D/A Configuration dialog box. The Hardware dialog box, shown in Figure 81, gives you the opportunity to select an output signal range from those available on your board. Use this range in conjunction with the gain setting in the D/A Configuration dialog box to most closely match the actual range of your output signal.

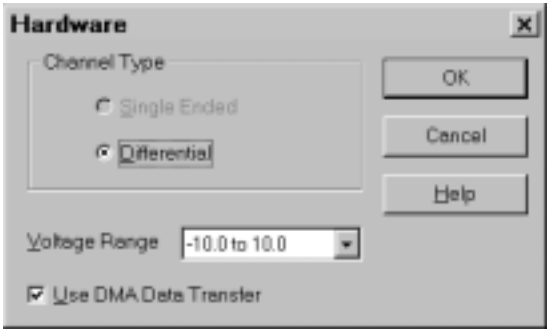

Figure 81: The Hardware Dialog Box

Also, select either the single-ended or differential output type.Your selection should match the actual type of output channel you intend to use.

Note: For information on the checkbox labelled “Use DMA Data Transfer,” see page 201.

Specifying a Subsystem Handle

When you are done configuring the analog output subsystem, click **OK**. A configured subsystem has an identifier, called a *subsystem handle*, which is used by other DT VPI objects to identify which subsystem to use. All data acquisition objects require a subsystem handle (hSubsys) as one of the data inputs.

Starting an Analog Output

There are several ways to start analog data output, and the easiest way is to do nothing! By default, the D/A Config object is set to start analog output automatically when a Put Data Panel or Put Data Function object executes. The Put Data Panel or Put Data Function object must have at least 0.5 seconds of data before it executes, or it raises an error.

Another way to start analog output is to use the Start Acq object (**DT DataAcq ⇨ Control ⇨ Start Acq**). This object is different than the regular Start object in that it starts a subsystem on the data acquisition hardware, not the execution of a VEE thread. If you do not want analog output to start automatically when the Put Data Panel or Put Data Function receives the data, include a Start Acq object after the Put Data Panel or Put Data Function object. Be sure to deselect the Automatic Start option in the D/A Advanced Configuration dialog box.

Note: If Automatic Start is selected in the D/A Configuration dialog box, executing a Start Acq object after a Put Data Panel or Put Data Function object raises error 2010.

Outputting Analog Data from a Single Channel

The simplest analog data output program outputs a specified number of samples from a single channel. Figure 82 shows what the DASIMPLE.VEE example program (in the \Examples\Dataacq directory) looks like.

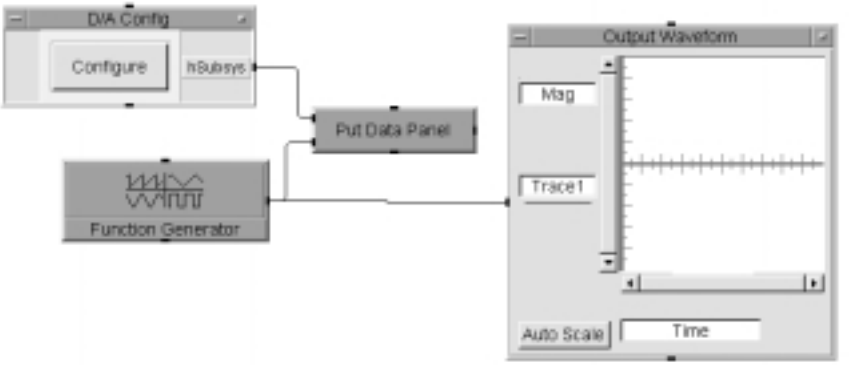

Figure 82: Example on Disk: DASIMPLE.VEE

As you learned in the last section, the D/A Config object is required. Click the **Configure** button, (select the board to use, if necessary), and specify the channel, gain, and sampling rate configuration. In this example, the default configuration of channel 0 at a sampling rate of 1000 Hz is used.

Take a moment to consider the connection from the D/A Config object to the Put Data Panel object. The Put Data Panel object is always to the right of the D/A Config object because it needs the subsystem handle from the D/A Config object.

In the Put Data Panel object, you must also indicate which channel you have selected in the D/A Configuration dialog box. Enter this information directly in the user edit box

---

Note: Make sure that each Put Data Panel object has a corresponding channel configured in the D/A Config object. Mismatched channels result in a run-time error.

A Function Generator object provides the data to output. It is set to generate 1000 points of a 10 Hz cosine wave over a time span of 1 s. Set the Rate in the D/A Configuration dialog box to 2000 Hz. Click **Add** (or **Update**) and then click **OK** to close the dialog box. Using these settings, compute the duration of the generated waveform from the DAC to make sure that the waveform has a duration of at least 0.5 s:

$$\frac{1000 points}{\frac{2000 samples}{1 second}} = 1000 points \bullet \frac{1 second}{2000 samples} = \frac{1}{2} second$$

Note: Refer to "Selecting an Output Rate," starting on page 150 for more discussion on setting up the Function Generator object to use with the DACs.

Since 0.5 seconds of data is the minimum amount of data needed by the Put Data Panel object to start the D/A subsystem, we have just enough data.

Run the program by clicking **Run**. Figure 83 shows what the program looks like.

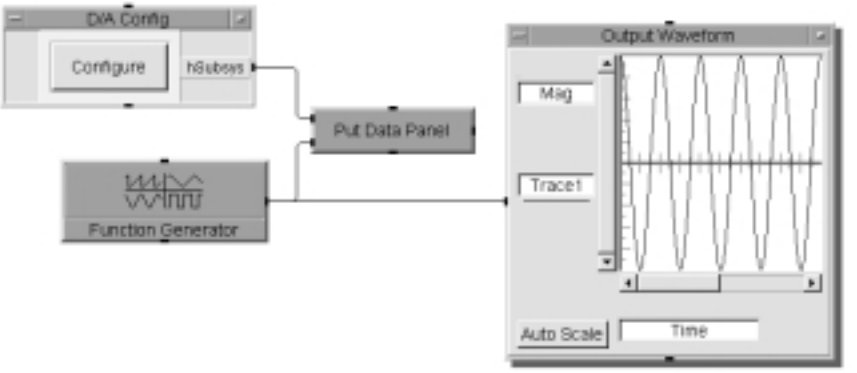

Figure 83: Example Program DASIMPLE.VEE after It Has Been Run

Outputting Analog Data Continuously

Outputting data continuously is a simple extension of the last program. You need a Put Data Panel object for each configured channel. For example, Figure 84 shows what the DACONT.VEE example program (in the \Examples\Dataacq directory) looks like.

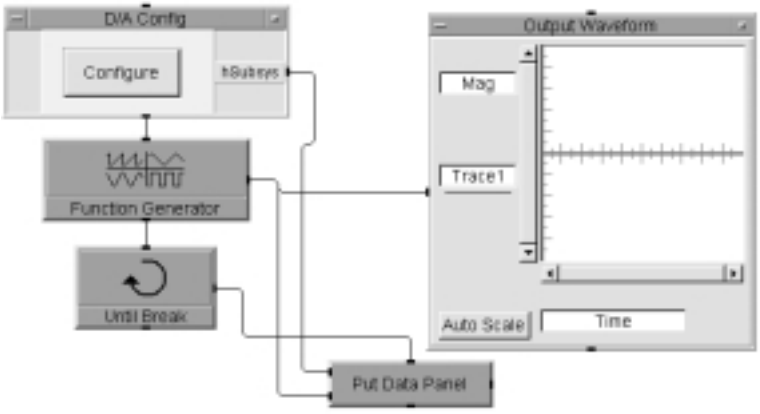

Figure 84: Example on Disk: DACONT.VEE

A one-to-one correspondence exists between the number of Put Data Panel objects and the number of channels selected in the D/A Configuration dialog box. For this example, the D/A Config object was configured only for channel 0, as shown in Figure 85.

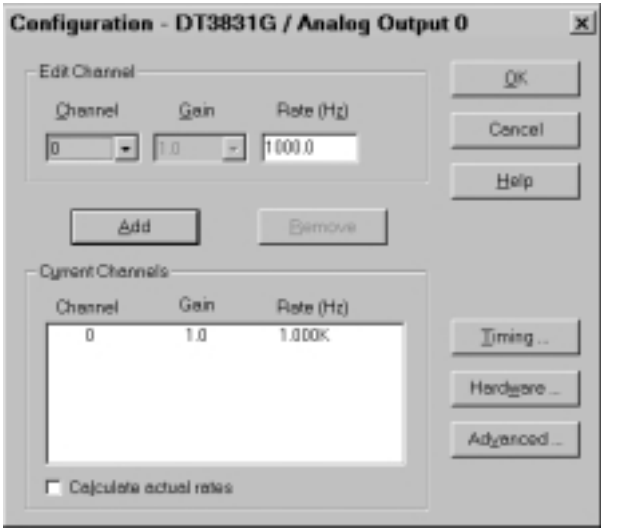

Figure 85: Configuration for One Channel

What's Really Going On?

Figure 86 illustrates how data is stored in memory and then retrieved by DT VPI.

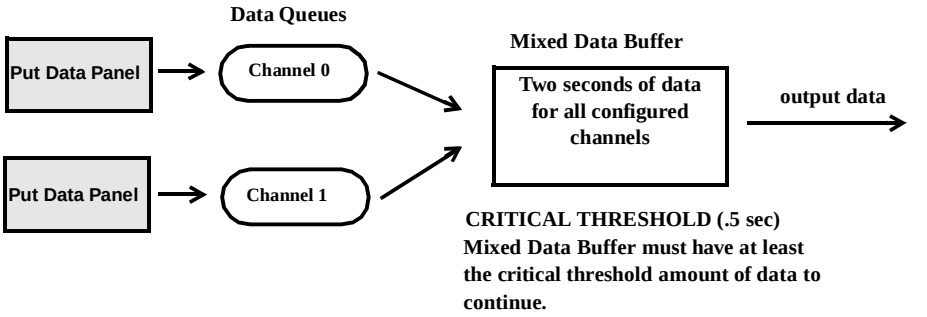
Data Queues
Mixed Data Buffer
Put Data Panel
Channel 0
Two seconds of data for all configured channels
output data
Put Data Panel
Channel 1
CRITICAL THRESHOLD (.5 sec)
Mixed Data Buffer must have at least the critical threshold amount of data to continue.

Figure 86: Acquired Data is Stored in Memory, then is Retrieved by DT VPI

Now, let's look at this program to learn the how DT VPI handles data:

- When you click **Run**, the D/A Config object initializes the board and sets up the channel list. Enough memory for two seconds of data for all channels is set aside as the *Mixed Data Buffer.* The subsystem is in Ready state.
- The Function Generator object generates 1000 points of data and passes the data to the Put Data Panel object. The Function Generator object generates data once only.
- When the Put Data Panel object executes (Automatic Start is checked in the D / A Configuration dialog box), the subsystem is in the Running state. The data for each channel (in this case, one channel) is stored in its own *data queue*. DT VPI sets aside enough memory for each data queue to hold four seconds of data, depending on the channel's sampling rate.
- On each iteration of the Until Break object, the data is pulled out of the data queue, mixed together (if another channel were configured), and stored in the Mixed Data Buffer. If one channel has less data than is allowed by the critical threshold, then, depending on the data mode for that channel, that channel goes to the Done state (in Gap-Free data mode) or to Waiting state (in Allow Gaps data mode).

  The goal is to keep the Mixed Data Buffer as full as possible. If, however, VEE is executing an object that takes awhile to complete, the Mixed Data Buffer might empty. If this happens, data acquisition stops on all channels. You can remedy this situation by priming the Put Data Panel object with more data.
- When the thread finishes executing, the program stops but data acquisition continues until the Mixed Data Buffer is emptied; then, data output stops. If you pause or stop the program (by clicking **Stop**), data output continues until the data queues and Mixed Data Buffer empty or until a Stop Acq object is executed.

States of a D/A Channel

The data queues and Mixed Data Buffer are filled by the Put Data Panel objects and emptied as the samples are output by the board. To check how much memory is being used by the samples, use the Acq Status object. The Acq Status object also provides the state of the channel. The possible states are listed in Table 7.

Table 7: States of a D/A Channel

Status from Acq Status Object	Channel Status
Ready	Subsystem is initialized but not started. There is no data in memory yet. Executing a Stop Acq or Set Config object returns all channels in the subsystem to Ready.
Waiting	The channel is waiting for data because **1.** The channel has data to output but another channel does not have data to output. The subsystem is waiting for all channels to have data before it can go to the Running state. **2.** If the data mode is Allow Gaps, the channel is experiencing a gap and is waiting for data to be available. The last value received is continuously output until new data is received.
Running	The hardware is outputting data on the channel or waiting for an external trigger.
Done	The channel Stopped for one of the following reasons: **1.** The data mode is Gap-Free and no more data is available. **2.** Burst mode is finished. **3.** Hardware errors, such as underrun (2007) or trigger (2006) error, occurred. **4.** Driver underrun error (2081) occurred. The data queue was not able to send data to the Mixed Data Buffer fast enough because an object was executing too long. Either change your program or raise the critical threshold.

The Critical Threshold

When a channel is configured for Gap-Free data mode, if at any time there is less than 0.5 seconds of data for that channel in the Mixed Data Buffer, that channel goes to the Done state. Similarly, when a channel is configured for Allow Gaps data mode and less than 0.5 seconds of data is available for that channel in the Mixed Data Buffer, DT VPI pads the data with up to 0.5 seconds worth. and that channel goes to the Waiting state. The value of 0.5 seconds is called the *critical threshold*. You can change this value by calling the function dtv.SetCriticalThreshold. See Chapter 10, "Advanced Functionality," in this manual.

Outputting a Single Analog Value

Outputting a single value of analog data is typically used when you are not concerned with a pacer clock. The Put Single Value object is used to output a single value. The DASINGLE.VEE example program (in the \Examples\Dataacq\ directory), shown in Figure 87, is a typical program using the Put Single Value object.

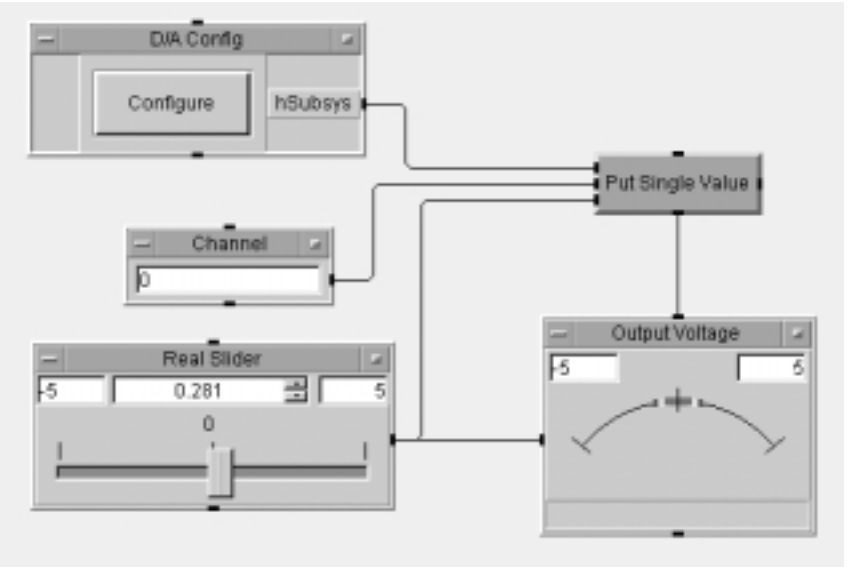

Figure 87: Example on Disk: DASINGLE.VEE

As usual, the D/A Config object configures the D/A subsystem on the data acquisition hardware. However, the Put Single Value object does not refer to the channel list defined in the D/A Configuration dialog box. You need only specify the channel number as an input to the Put Single Value object (usually in the from of an Int 32 Constant object).

The subsystem handle is passed to a Put Single Value object. In addition to the subsystem handle, the Put Single Value object needs to know which channel to use for output. An Int 32 Constant object, which we have retitled as Channel, provides this information. When you run the program, as soon as it receives the data, the Put Single Value object outputs one value to the specified channel.

Outputting a Burst of Data

If you know exactly how many samples of data you want to output or the time period over which you want to output, use *burst mode*. The DABURST.VEE example program (in the \Examples\Dataacq\ directory), shown in Figure 88, shows a simple program.

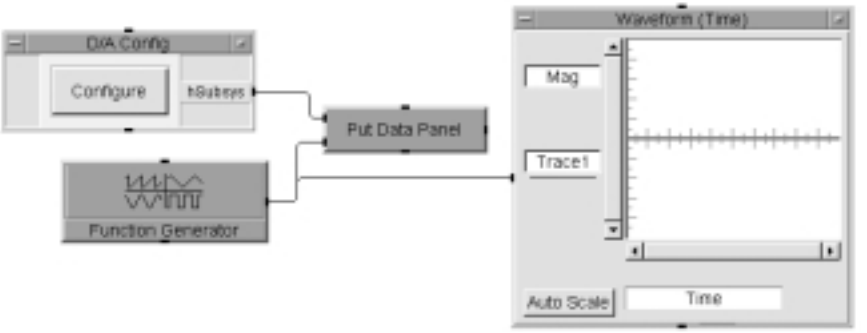

Figure 88: Example on Disk: DABURST.VEE

Configuring for Burst Mode

You probably noticed that the program to output data in burst mode uses the same objects we have already worked with. The difference lies in the configuration. To set up burst mode, you need to use the D/A Advanced Configuration dialog box. On the D/A Config object, click **Configure**. Because you intend to output a burst of data, click **Advanced**. The D/A Configuration dialog box enlarges to show advanced features, one of which is burst mode.

Using the D/A Advanced Configuration dialog box, you can set up the channel list just as you would normally. For this example, we intend to use channel 0 only. Click the entry for channel 0 in the Current Channels section. Select the checkbox labeled Enable in the Burst Duration section, and the radio button next to Samples.

Channel	Rate (Hz)	Gain	Burst Mode Enabled?	Samples
0	2000	1	yes	1000

Because the Function Generator object outputs 1000 points in 1 second, we need the D/A subsystem to output at a rate of 2000 samples per second, or 2000 Hz. Because the Function Generator is generating 1000 points, enter 1000 in the edit box next to Samples. (You could also type in 1 and select Seconds.) Click **Update**.

Note: Refer to "Selecting an Output Rate," starting on page 150 for more discussion on setting up the Function Generator object to use with the DACs.

The dialog box should look like that shown in Figure 89 when you are done.

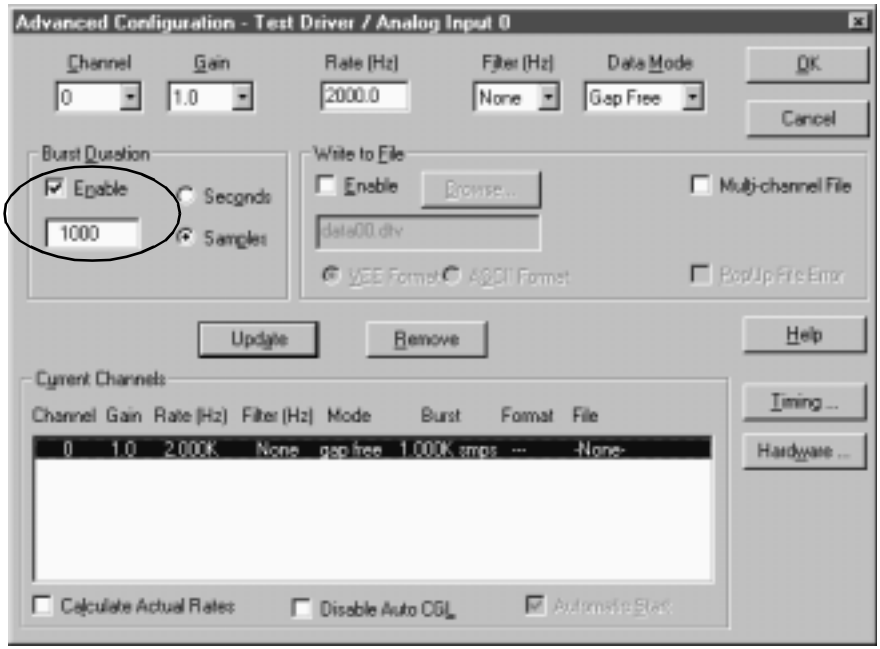

Figure 89: Configuring the D/A SubsystemTo Output Data in Burst Mode

Click **OK** to close the dialog box. The program is ready to run.

What's Really Going On?

When you run this example, the D/A Config object executes first, configuring the D/A subsystem on the board for burst mode. Only enough memory is set aside for the amount of data specified in the burst mode section of the D/A Configuration dialog box. Meanwhile, the Function Generator object is generating 1000 points of a 10 Hz cosine wave. It passes the data to the Put Data Panel object. When execution passes to it, the Put Data Panel object stores the data in memory and automatically starts the D/A subsystem. Then, the board outputs the data on the specified channel.

Note: Because only enough memory is set aside for the specified number of samples or number of seconds of data, using burst mode results in more efficient use of memory.

Changing the Subsystem Configuration

The **DT DataAcq** menu provides objects to check and change the current subsystem configuration programmatically. In previous examples, you specified the desired configuration using the subsystem configuration object (A/D Config, D/A Config, Digital In Config, or Digital Out Config), before you ran the program.

Two types of configuration objects are available:

- Those that obtain (or "get") the values of certain parameters for a given subsystem, and
- Those that change (or "set") the values of certain parameters.

Like the other data acquisition objects, the set and get configuration objects require a subsystem handle as one of the data inputs. Table 8 summarizes the configuration objects and their purposes.

Table 8: Summary of the Get and Set Config Object s

Name	Description
Get and Set Sampling	Obtain or specify the sampling rate and gain settings for each channel in a given subsystem.
Get and Set Clocking	Obtain or specify the clocking source and external clock frequency for a given subsystem.
Get and Set Channel Buffer Size	Obtain or specify the buffer size of a given channel for a given subsystem.
Get and Set Range	Obtain or specify the programmable voltage range for a given subsystem.
Get and Set Trigger	Obtain or specify the trigger source for a given subsystem.

Table 8: Summary of the Get and Set Config Objects (cont.)

Name	Description
Get and Set Sample Clock Frequency	Obtain or specify the channel-to-channel sample frequency for an internal re-trigger and triggered scan of channels.
Get and Set Channel Filter	Obtain or specify the low-pass (cutoff) frequency for a given channel on a given subsystem.
Get and Set Data Mode	Obtain or specify the data I/O management configuration for the given channels for a given subsystem.
Get and Set Multi-Channel Mode	Obtain or specify the use of multi-channel file output, aggregate rate, and automatic build of the channel-gain list.
Get and Set C/T Mode	Obtain or specify the mode, gate source, gate type, and pulse type of a C/T subsystem.
Get and Set C/T Frequency	Obtain or specify the frequency and duty cycle of a C/T subsystem.
Get and Set C/T Pulse Width	Obtain or specify the delay and pulse width of a C/T subsystem.
Get and Set C/T Measure Duration	Obtain or specify the duration over which to measure events.

Note: For more information on the get and set config objects, refer to the DT VPI online help.

Often, the get and set config objects are used as a pair. Place the get config object so that it executes before the set config object. Then connect the outputs from the get config object to the inputs of the set config object, except for the one or two parameters you intend to change.

CAUTION:

Because the set config objects change the configuration of the subsystem, it is important that you not reconfigure a running subsystem.To avoid this situation, place a Stop Acq object before the get config object.

For example, the program shown in Figure 90 uses the Get Sampling and Set Sampling objects to change the analog output rate (the sampling rate). The D/A Config object is configured to output a 1000 Hz on channel 0 (DAC0). Automatic Start is selected (as usual). We have retitled the Allocate Real 32 Array Object to Real Array w/ New Rate.

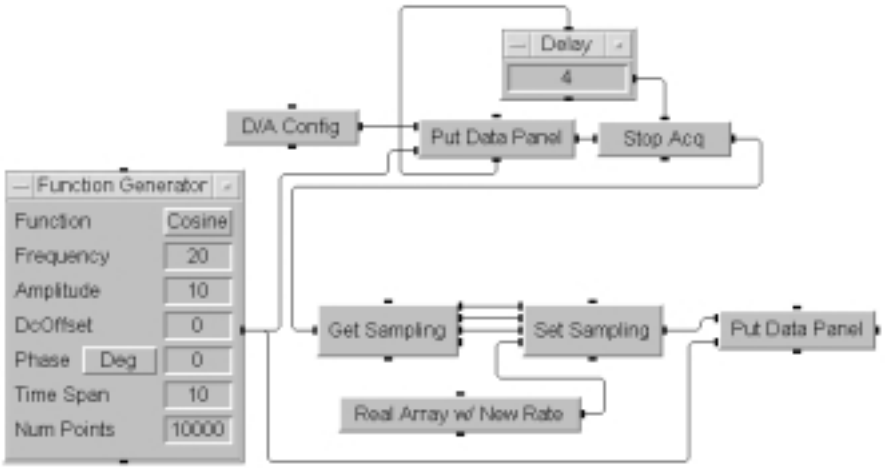

Figure 90: A Program To Change the Output Rate Programmatically

When you run this program, 10,000 points (10 seconds) from the 20 Hz cosine signal generated by the Function Generator object are passed to the first Put Data Panel object. Since Automatic Start is selected, the Put Data Panel object passes the data to the D/A

subsystem, which immediately starts outputting the data.

The Put Data Panel object finishes executing, even though all the data has not yet been output from the board. The Delay object, shown in Figure 91, allows the hardware some time to output the data before the subsystem is stopped by Stop Acq. Alternatively, the Acq Status object can be used to verify when all the data has been output.

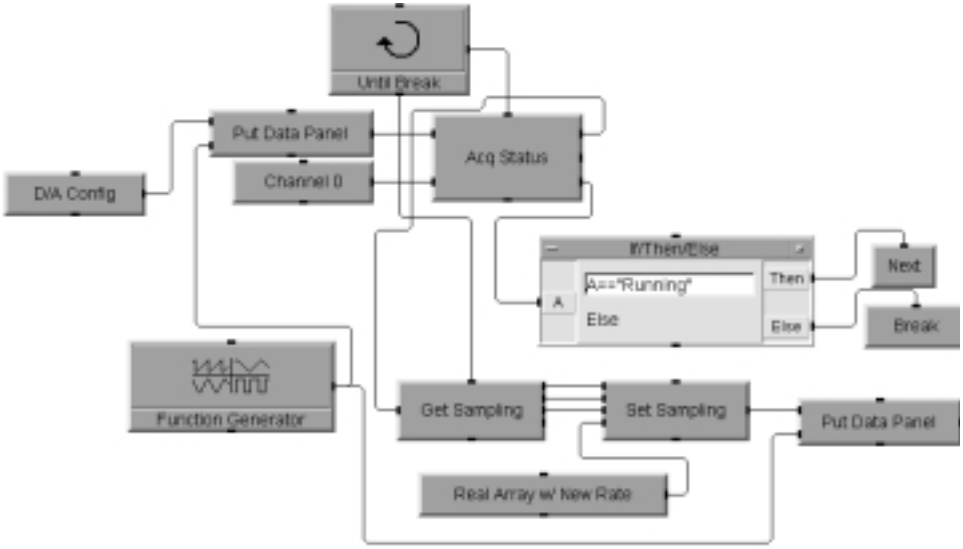

Figure 91: The Same Program Using Acq Status instead of Delay

The Get Sampling and Set Sampling objects, used as a pair, allow a change to the D/A sampling rate. Because the Set Sampling object expects the Rate input as a real array containing one value for each channel, the new rate is in an Allocate Array object. We have retitled this object as Real Array w/ New Rate. This object allocates a one-dimensional array containing only one value (we are using only one channel), which is the new sampling rate in hertz. We have also retitled the Int 32 object as Channel.

When the second Put Data Panel object receives the same data and a handle to a reconfigured D/A subsystem, the Put Data Panel object outputs the data using the new sampling rate. Because Automatic Start is selected in the D/A Configuration dialog box, a Start Acq object is not needed.

Note: If you use Start Acq with Put Data Panel, place Start Acq *after* the Put Data Panel object and be sure to deselect Automatic Start in the D/A Configuration dialog box.

Handling Acquisition Errors

In general, if an error occurs during acquisition, a dialog box appears and your program stops running. To clear the dialog box, click **OK**. Any data in memory at the time is flushed out. You need only click **Run** or the Start object to restart your program. For example, if a Get Data Panel or Get Data Function object tries to retrieve data for a channel for which there are not enough samples available, an error is raised (see Figure 92). Similarly, if the Put Data Panel or Put Data Function object executes before it has at least 0.5 seconds of data (the critical threshold), an error is raised. In both cases, the subsystem is no longer running, so a Stop Acq object is not needed.

Error Message
The Acquisition/Output is complete for the channel. This is typically caused by processing that can not keep up with the acquisition or output rate.
in Compiled Function 'GetData'
Object Title: GetDataPrimitive
Object Type: Call
Error number: 2012
OK

Figure 92: Error Raised When Not Enough Data Is Available in Memory

Errors When Reading From File

When reading data directly from a file and outputting it, you want to be sure of the data's integrity. Therefore, you want to know of any error that occurs immediately —whether it be an output error, a read error, a problem with the file, or whatever.

You can monitor file errors in one of two ways. One way is to enable PopUp File Error detection in the Read from File edit box. Enabling this feature permits errors to appear immediately in a pop-up dialog box. This error cannot be trapped. The other way is to use the Acq Status object to repeatedly poll the subsystem to determine if file output stopped for any reason.

Trapping Errors

If you wish to trap the errors for other purposes, such as logging them for a report, you need to add an Error output pin to the data acquisition object. However, because most data acquisition objects are secured UserObjects, you cannot directly add an error terminal as you can to other VEE objects. To verify this fact, try to select **Add Terminal** from Put Data Panel's object menu. Notice that it is grayed out.

To add an error output pin to a data acquisition object, select the object and put it into a UserObject (**Edit** ⇨ **Create UserObject**). Then, select the UserObject's object menu ⇨ **Add Terminal** ⇨ **Error Output**. The error output pin is usually connected to the sequence in pin of a Stop Acq or Close Subsystem object, if present.

See the Analog Input example "Trapping Errors," starting on page 122, for more information.

Reading Data From a Disk File

As you learned in the section "Sending Data to a Disk File," starting on page 125, you can send acquired data to disk in one of two ways. One way is to use the VEE To File object. This data can be read in with a From File object (**I/O ⇨ From ⇨ File**). The data can then be used with a Put Data Panel or Put Data Function object.

If you need to read and output data very quickly (over 50 kHz), DT VPI provides the capability to read the data directly from a disk file and pass it immediately to the DAC. The file is treated as though it were a data queue. The binary file contains a special file header, which includes all the information necessary to convert the raw data when it is output. The file header is described in Appendix C of this document.

The DAFILE.VEE example program (in the \Examples\Dataacq\ directory) shown in Figure 93, creates a file of data on disk. Then, the D/A subsystem reads the data, and directly outputs it to the D/A subsystem.

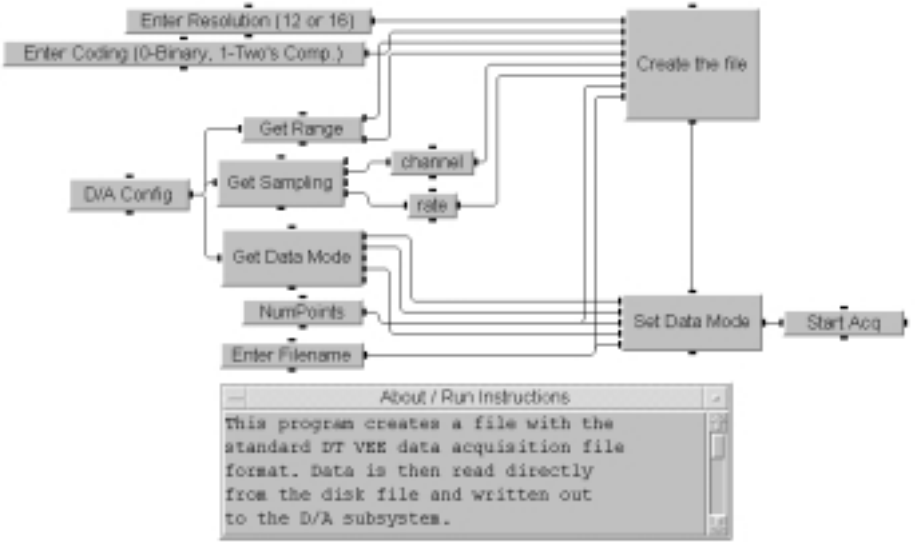

Figure 93: Example on Disk: DAFILE.VEE

Although the example program DAFILE.VEE configures the D/A subsystem programmatically, you need to know how to use the D/A Advanced Configuration dialog box. Set up the channel list just as you would normally. As you add each channel, click the checkbox labeled **Enable** in the Read from File section. Set the channels to the following rates and gains:

Channel	Rate (Hz)	Gain	Read from File Enabled?	File Name
0	10000	1	yes	data00.dtv

When this program is run, the D/A subsystem is configured. The program collects information from the user, such as the resolution and encoding of the board, the range, channel, rate, number of points, and the name of the file to create. The UserObject Create the file creates the file on disk containing the special file header and raw data ready to be output to the D/A subsystem. The Set Data Mode object essentially finishes the configuration that the D/A Config object started.

When the Start Acq object is executed, the D/A subsystem reads the file and outputs the data directly and quickly.

If your board supports Gap Free data with single DMA, you can continuously output a file of data to a DAC. Configure the DAC for Allow Gaps data mode and enable Read from File.

Receiving Data From DDE

Dynamic Data Exchange, or DDE, defines a message-based protocol for communication between Windows applications. The communication takes place between a *DDE client* and *DDE server*. The DDE client, (in this case VEE), requests a conversation with the DDE server about a certain topic. The client can then request information or services from the server, send information to the server, or wait. The server responds by sending data or executing procedures.

Earlier in this chapter we looked at sending data to a DDE server. In the example shown in Figure 94, we will read the data back in to a VEE program. The Put Data Panel object sends the data for output to an analog device.

Figure 94: A Program to Receive Data through DDE

In the To/From DDE object (**I/O ⇨ To/From DDE (PC)**), by default the Application is Excel and the Topic is Sheet1, an Excel document. Double-click the empty horizontal bar below Sheet1 and fill in the information for the transaction as shown in Figure 95.

Figure 95: DDE Transaction Settings To Read Previously-Acquired Data from Excel

Leave READ (REQUEST) as is. Click **STRING FORMAT** and change it to TOKEN FORMAT. Click **SCALAR** and change it to 1D ARRAY, then click **SIZE** to change it to TO END.

When working with DDE and using VEE as the client, you typically need to find out how the server application expects the item in the conversation to be phrased. The Item is a string specifying the starting row and column and the ending row and column from which the data should be read. A colon (:) separates the starting and ending points. For example, if you have 100 values in one column, from row 1 to row 100, the Item is specified as “r1c1:r100c1”. In the TEXT edit box, enter the name of the one-dimensional array to receive the data. (You also need to add an output terminal for the data.)

What's Really Going On?

When you run this program, a conversation with Excel is initiated by the To/From DDE object. If it can find the server application, VEE launches the server application. Then, the data is read from Excel, from row 1, column 1 to row 100, column 1. The data is passed to the Put Data Panel object, and then output from the D/A subsystem.

Example on Disk: The DDE.VEE example program (in the \Examples\Dataacq\ directory) uses Microsoft Excel as the server application. 512 samples of data are acquired and written to an Excel file SHEET1.XLS in one column.

Optimizing Analog Output

You can optimize digital-to-analog conversions in DT VPI by doing the following:

- Use burst mode whenever appropriate.
- When outputting data continuously, use the Allow Gaps data mode when it is not necessary to output every single value. For example, when outputting an acquired music signal, Allow Gaps would not be appropriate, but it might be appropriate when outputting data to run a robot arm. When a gap of data is encountered by Put Data Panel or Put Data Function, the last data element received is repeatedly output until more data element is present. Allow Gaps is found in the D/A Advanced Configuration dialog box. See the DT VPI online help for more information.
- When using a relatively slow output rate, use the Acq Status object to check on how many data points are waiting to be written to the channel's data queue. If the Put Data Panel or Put Data Function object attempts to write data to a full data queue (and the D/A subsystem is configured for Gap Free data mode), error 2012 is raised.
- When configuring, leave Calculate Actual Rates unchecked until you are done creating the channel list. Doing so avoids having DT VPI take the time to calculate between each channel entry.
- Use the Close Subsystem object after your data has been output to release system resources such as memory, DMA channels, and interrupts.
- Use the Put Data Function object instead of the Put Data Panel object. The Put Data Function object runs more efficiently because it is a function call, not a UserObject.

Digital Input and Output

Connecting External Devices 187
Configuring for Digital I/O 187
Acquiring a Single Digital Value 189
Outputting a Single Digital Value 191

To communicate with a device that is itself digital, many data acquisition boards provide *digital input* (DIN) and *digital output* (DOUT) subsystems.

This chapter discusses how to control your data acquisition hardware to perform digital input and output. Configuration is simple—you just specify whether you want to use the DIN subsystem, the DOUT subsystem, or both. Channel lists and sampling rates are not used in digital input and output.

This chapter also includes many example programs and explanations of using the data acquisition objects provided by DT VPI.You will learn how to create programs to

- Input digital values repeatedly; and
- Output digital values repeatedly.

Connecting External Devices

Using the screw terminal panel's digital lines (also called Bit 0, Bit 1, Bit 2, etc.), connect your digital input or output device. Refer to your hardware user manual for more information on connecting external devices.

Configuring for Digital I/O

Typically, your data acquisition board has between two and 16 digital lines. Check your hardware user manual to find out how many digital lines your board has and how they are organized as ports. If they are organized as a single digital port, then you can configure the digital lines as either digital input or digital output, but not both at the same time. If they are organized as more than one port, then you can configure one port for digital input and another port for digital output, if you wish, or all ports as the same type.

Note: To find out the current organization of the ports, use the advanced function dtv.GetResolution in a Call Function object (see page 237). To change the organization of the ports, use the function dtv.SetResolution.

Select a Digital In Config object to specify that you want to use all available digital lines or one port for input. Select a Digital Out Config object to specify that you want to use all available digital lines or one port for output.

When you click the **Configure** button on a Digital In Config or Digital Out Config object, you do not see a dialog box to set up a channel list. A channel list is inappropriate when working with digital input and output devices. Instead, the Select Device and Subsystem dialog box appears, as shown in Figure 96.

If you have more than one device installed in your computer, then select a device from the drop-down list under Device Name. Once a board is selected, DT VPI indicates whether the digital lines are organized as one port or more than one port. If they are organized as one port, then “DIN Unit 0” appears grayed out under SubsystemTo Use. However, if they are organized as more than one port, then DIN Unit 0 is not grayed out, and you can choose among the ports.

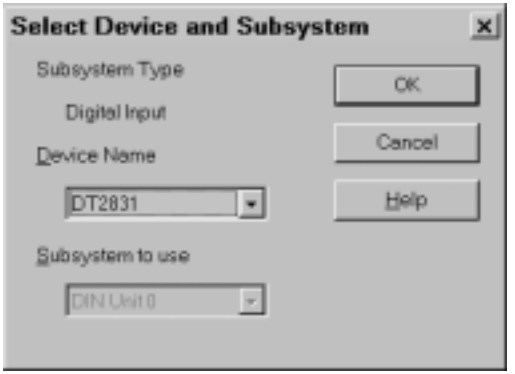

Figure 96: The Select Device and Subsystem Dialog Box

Acquiring a Single Digital Value

Digital data acquisition is accomplished one value at a time, using a Get Single Value object for each digital port. First, you need a Digital In Config object (**DT DataAcq ⇨ Digital In Config**). This object configures the hardware for digital input and defines a subsystem handle. We have retitled the Int 32 object as Channel, the User object as DinDemux, and the Alphenumeric objects as Bit Number objects one through eight. The DIN.VEE example program (in the \Examples\Dataacq\ directory), shown in Figure 97, continuously acquires digital data from a digital device. It can be used to verify the state of each bit in the digital input value.

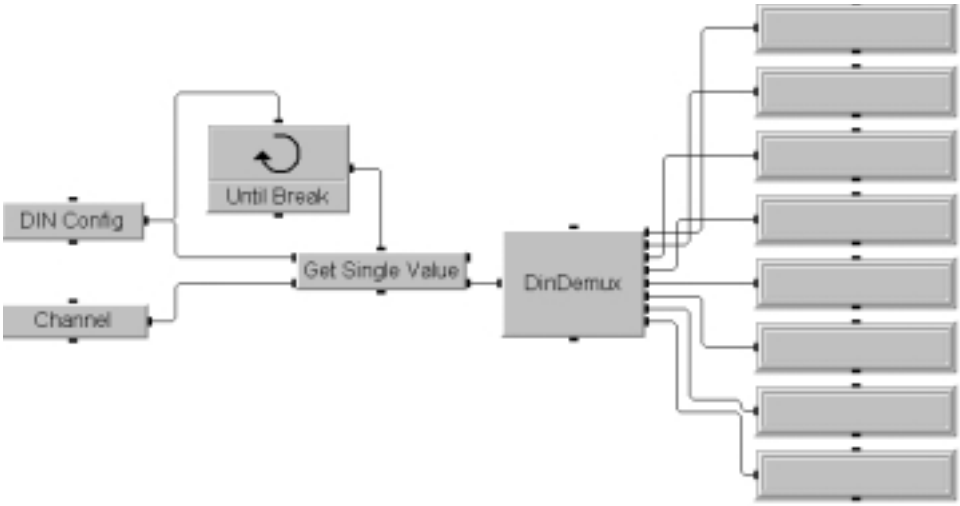

Figure 97: Example on Disk: DIN.VEE

When you run this program, a value on each connected digital line is immediately acquired. This value is masked in the UserObject DinDemux, revealing the state (0 or 1, LOW or HIGH) of each bit in the value.

Note: If you want to change the bit resolution of the DIN subsystem and your board supports this option, you can use the advanced function dtv.SetResolution. See Chapter 10, "Advanced Functionality," for information on this function.

Example on Disk: The DIO.VEE example program (in the \Examples\Dataacq\ directory) demonstrates continuous acquisition and output of digital data. The user selects a value to output using the slider, and the meter displays the acquired digital value.

Outputting a Single Digital Value

Outputting a single digital value is similar to acquiring a single value, except that you use a Put Single Value object instead of a Get Single Value object. You use one Put Single Value object for each digital port. The Digital Out Config object (**DT DataAcq ⇨ Digital Out Config**) configures the hardware for digital output and defines a subsystem handle. We have retitled the Int 32 object as Channel and the Formula object as DoutMux. The DOUT.VEE example program (in the \Examples\Dataacq directory), shown in Figure 98, continuously outputs user-selected digital values to a digital device.

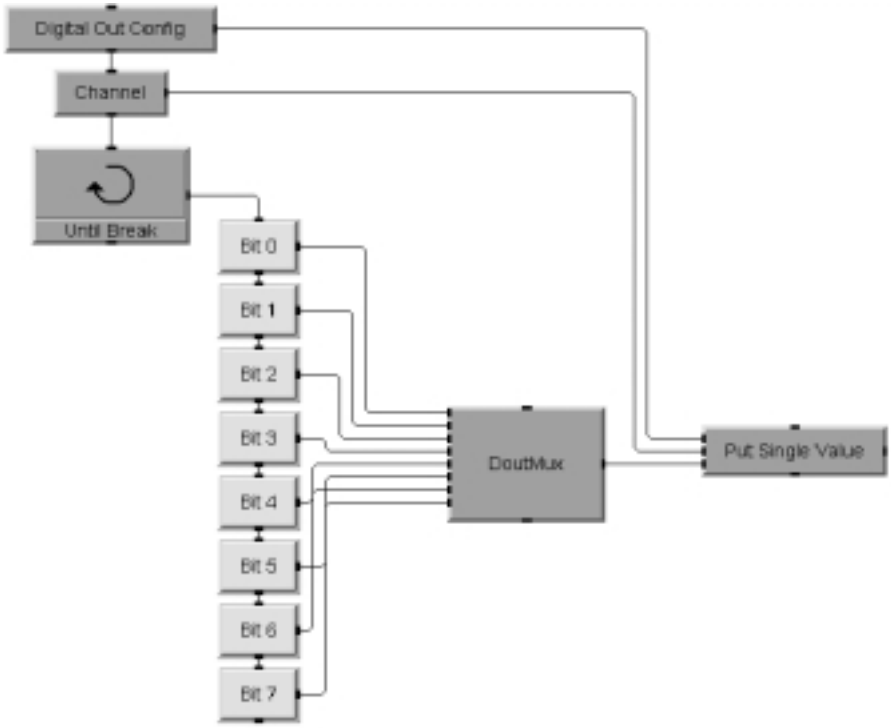

Figure 98: Example on Disk: DOUT.VEE

After configuring the hardware for digital output, select the output value by clicking the toggle buttons. One button exists for each digital line. Depressing the button sets the corresponding bit to a logical 1. Leaving the button up sets the corresponding bit to logical 0. When you run this program, the values of each bit are added together in the UserObject DoutMux, resulting is a decimal value. This value is passed to the Put Single Value object and output to the digital device.

Example on Disk: The DIN.VEE example program (in the Examples\Dataacq\ directory) demonstrates continuous retrieval of single values from a digital input subsystem.

Note: If you need to change the resolution of the DIN or DOUT subsystem and your board supports this option, you can use the advanced function dtv.SetResolution. See Chapter 10, "Advanced Functionality," for information on this function.

Simultaneous A/D and D/A

Hardware Support 195
Starting Two Subsystems Simultaneously 200
Sharing DMA Resources 201

This chapter provides device-specific information about running the A/D and D/A subsystems at the same time. You will learn

- Whether your hardware supports running two subsystems simultaneously, and if so, using which data transfer modes;
- How to start two subsystems simultaneously;
- How to prime the analog output subsystem with data; and
- Considerations for using the computer's DMA resources.

Hardware Support

Table 9 lists the data acquisition boards from Data Translation and the *combinations* of data transfer methods that are supported when using the A/D and D/A subsystems simultaneously.

The data transfer methods are DMA, Bus Master, Single Value, and Polled I/O. For analog input, you are using the *DMA* transfer method when you use the Get Data Panel or Get Data Function object, and check "Use DMA Data Transfer" in the Hardware dialog box (see page 87). You are using *Polled I/O* or *PCI Bus Mastering* if you leave "Use DMA Data Transfer" unchecked in the Hardware dialog box (default). *Single Value* is performed by the Get Single Value object.

For analog output, the situation is similar.You are using the DMA transfer method when you use the Put Data Panel or Put Data Function object, and check "Use DMA Data Transfer" in the Hardware dialog box (see page 153). You are using Polled I/O if you leave "Use DMA Data Transfer" unchecked in the Hardware dialog box (default). Single Value is performed by the Put Single Value object.

To read the table, first locate your board in the left column. For each board, at least one row exists, with each row containing a pair of stars (*). From each pair, one star is in one of the Analog Input Subsystem columns and one star is in one of the Analog Output Subsystem columns. Together, the two stars indicate a valid combination of transfer modes when using the analog input and output subsystems simultaneously in a VEE program using DT VPI.

For example, the row for the DT01-EZ board contains only one pair of stars: Analog Input Single Value and Analog Output Single Value. Therefore, if you are using the DT01-EZ board, you can use the analog input and output subsystems simultaneously only when using the analog input subsystem with the Get Single Value object, and the analog output subsystem with the Put Single Value object.

Table 9: Hardware Support for Simultaneous A/D and D/ A

Boards	Analog Input Subsystem				Analog Output Subsystem		
	DMA	Bus Master	Single Value	Polled I/O	DMA	Single Value	Polled I/O
DT2801, DT01-EZ			*			*	
DT2812			*			*	
			*				*
DT2821 Series, DT21-EZ	*					*	
			*			*	
			*				*
DT2831 Series (excluding the DT2839)	*					*	
	*						*
			*		*		
			*			*	
			*		*		
				*			*
				*		*	
				*			*
DT2839	*				*		
	*					*	
	*						*
			*		*		
			*			*	
			*				*
				*	*		
				*		*	
				*			*

Table 9: Hardware Support for Simultaneous A/D and D/A (cont.)

Boards	Analog Input Subsystem				Analog Output Subsystem		
	DMA	Bus Master	Single Value	Polled I/O	DMA	Single Value	Polled I/O
DT31-EZ	* * *		* * *	* * *	* * *	* * *	* * *
DT301, DT302, DT303, DT304, DT321, DT322		* *	* *			* *	* *
DT3001, DT3003, DT3004, DT3005, DT3001-PGL, DT3003-PGL			* *	* *		* *	* *
DT3010, DT3010/32, DT3016		* *	* *			* *	* *
DT7102			* *	* *		* *	* *
DT9801, DT9802, DT9803, DT9804, DT9805		* *	* *			* *	

Notes: The DT23-EZ, DT24-EZ, DT24-EZ-PGL, DT301, DT303, DT321, DT3002, DT2814, and DT7101 do not have any D/A subsystems.

The DT2815 has no A/D subsystem.

The DT2839 and DT31-EZ feature single-channel DMA for the A/D subsystem, leaving a DMA channel available for the D/A subsystem.

The example program SIMULTAN.VEE (in the \Examples\Dataacq\ directory), shown in Figure 99, illustrates how to accomplish simultaneous A/D and D/A. Let us look at this program in more detail.

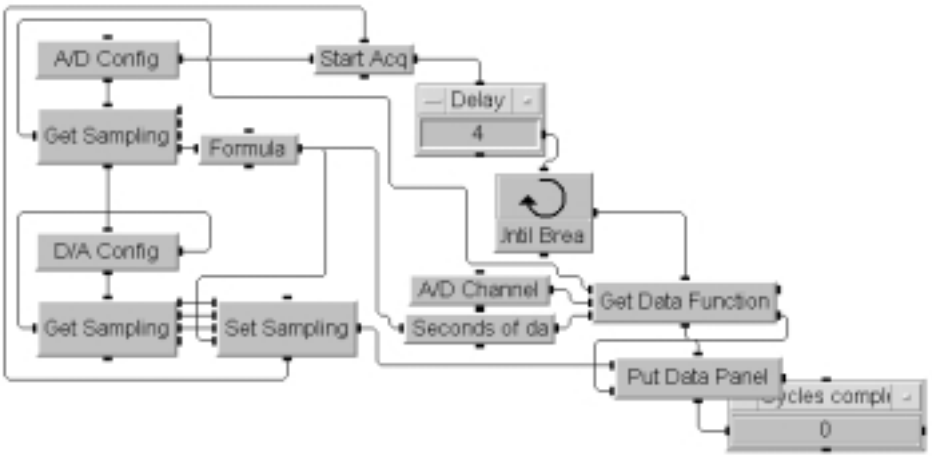

Figure 99: Example on Disk: SIMULTAN.VEE

In this program, the A / D subsystem and D / A subsystem run continuously and simultaneously, but they do not start at the same time. First, the two subsystems are configured for one channel, at a rate of 1000 Hz. To make sure that the two subsystems are set to the same rate, the D / A subsystem is set to run at the rate of the A / D subsystem, using the Get Sampling and Set Sampling objects.

Acquisition begins when the Start Acq object executes, and four seconds' worth of data (Delay is set to 4) is acquired and stored before the Get Data Function object retrieves it from memory. The delay is used as a precaution, ensuring that plenty of data is available for the Put Data Panel object. Every time through the loop, the Get Data Function object retrieves two seconds' worth of data, and passes it to the Put Data Panel object. (Two seconds' worth of data equals 2*rate.) The Put Data Panel object sends the data to memory; the data is then output from the board. The Counter object logs the iteration, and the loop begins again. Because Get Data Function retrieves the data before Put Data Panel puts it, and the data is continuously being acquired, the A / D subsystem is always ahead of the D / A subsystem.

Note: When performing simultaneous analog input and output, it is important that both subsystems be configured for the same sampling rate. Different sampling rates usually result in an "Acquisition Stopped" error (2012).

Starting Two Subsystems Simultaneously

When you run analog or digital input simultaneously with analog output, you always have to ensure that at least 0.5 seconds of data is available for the analog output subsystem before it begins to output. Therefore, it is best to start the analog input subsystem at least 0.5 seconds before the analog output subsystem starts.

However, you can start the Counter/Timer subsystem (C/T) with any of the other subsystems, simultaneously without cause for worry. Recall that the Start Acq object normally takes a single subsystem handle as input. Using a Formula object, create an array of subsystem handles and pass the array to the Start Acq object, as shown in Figure 100.

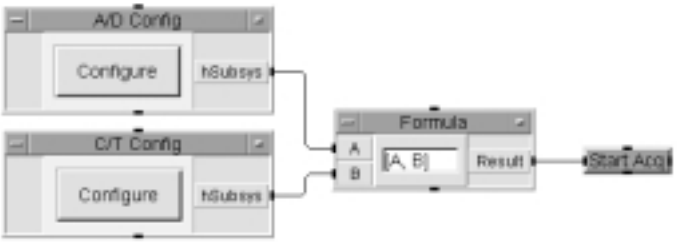

Figure 100: Starting Two Subsystems Simultaneously

Sharing DMA Resources

For ISA or EISA boards, *DMA* (Direct Memory Access) is a system resource you should consider when running two subsystems simultaneously. Some ISA or EISA boards can provide gap-free acquisition using only one DMA channel, others need two DMA channels. You indicate which channels to use when you install the board in your computer. Depending on the subsystem and the data acquisition board, DT VPI uses the fewest number of DMA channels possible, while still acquiring gap-free data. However, DMA resources are limited by the computer itself. Even with two boards in your system, you have at most three DMA channels: one board can use two DMA channels; the other board can use one DMA channel.

Some ISA boards use two DMA channels for the analog input subsystem. Therefore, if you want your program to run analog input and analog output simultaneously, consider the following recommendations:

- Disable DMA on the analog output subsystem. (Click **Hardware** from the D/A Configuration dialog box and uncheck the box next to "Use DMA DataTransfer.")You will not be able to run as fast as you would using DMA data transfer.
- Use two boards, making sure that one board is configured to use only one DMA channel, and that it does not use a channel that is already being used by the other board. Refer to your device driver documentation for information on configuring your board and driver.
- If DMA is more important than running the subsystems simultaneously, run the two subsystems sequentially, executing a Close Subsystem object in between.

Note: Refer to the device driver documentation accompanying your data acquisition hardware to find out if the analog input and output subsystems support gap-free data transfers with zero, one, or two DMA channels.

If your board supports OLSSC_SUP_GAPFREE_NODMA, you can run all subsystems simultaneously without using any DMA resource (leaving them available for other devices in your system). If your board supports OLSSC_SUP_GAPFREE_SINGLEDMA for A/D or D/A, you can run that subsystem with one DMA channel. If it supports OLSSC_SUP_GAPFREE_DUALDMA for A/D or D/A, you need two DMA channels to run that subsystem in continuous gap-free mode.

Using Counters and Timers

Terminology 205
Using the Counter / Timer as a Clock 209
Using the Counter / Timer as a Trigger 212
Using the Counter / Timer to Count Events 216
Using the Counter / Timer to Measure Frequency 218
Optimizing Counter / Timers 221

Aside from the pacer clock used internally by your data acquisition board (see Chapter 4), some boards provide user clocks, or *counter/timers*, for external use.

Counter/timers can be used in many configurations. Counters can be used singly or in combination with other counters. They can also be used with external or internal frequency clock sources and gates. The following diagram illustrates the components of a single counter/timer:

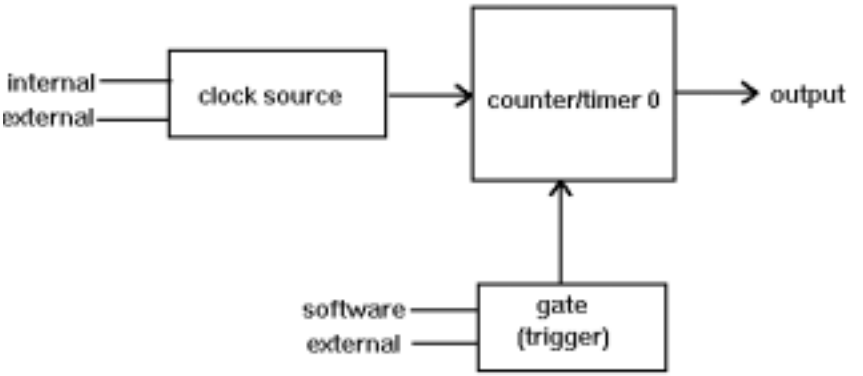

Figure 101: Components of a Single Counter/Timer

This chapter discusses how to control the counters/timers on your data acquisition hardware. This chapter includes many example programs and explanations of using the data acquisition objects provided by DT VPI. You will learn how to create programs to

- Use the counter/timer as a clock;
- Use the counter/timer as a trigger;
- Provide an external trigger to the counter/timer;
- Use the counter/timer to count events; and
- Use the counter/timer to measure frequency.

Terminology

Before you configure the counter / timer, you need to understand the terminology used when dealing with counter / timers.

In DT VPI, counter / timers can be set to one of four modes:

- *Generate Rate* mode to use the counter / timer as a clock.
- *One-shot* mode to use the counter / timer as a trigger.
- *Count Events* mode to use the counter / timer as a counter.
- *Measure Frequency* mode to use the counter / timer to measure the frequency of occurrence of an event.

Pulse Type

If the pulse type is *High-to-Low*, the low portion of the total clock period is the active portion of the counter / timer output. If the pulse type is *Low-to-High*, the high portion of the total clock period is the active portion of the counter / timer output.

Duty Cycle

The duty cycle indicates the percentage of the total clock period that is active. By default, the total clock period is divided evenly between high and low (50%). In Figure 102, the pulse type is Low to High and the duty cycle setting is 30%.

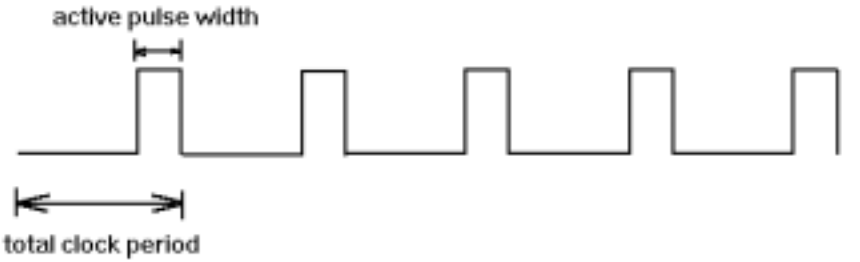

Figure 102: A Signal Generated by the Counter/Timer with a Low-to-High Pulse Type and a Duty Cycle Setting of 30%

Gate Types

If the gate source is *None*, the counter / timer operation is enabled immediately in software. If the gate source is *External*, one of the following gate types may be available.

- **High Level**: enable the counter / timer operation when the gate signal is high; disable the counter / timer operation when the gate signal is low.
- **Low Level**: enable the counter / timer operation when the gate signal is low; disable the counter / timer operation when the gate signal is high.
- **Rising Edge**: start and the continue counter / timer operation on the rising edge of the gate signal.
- **Falling Edge**: start and continue the counter / timer operation on the falling edge of the gate signal.

Figure 103 shows a signal generated by a counter / timer operation using a low-to-high pulse type and a duty cycle of 30%, enabled by an external high-level gate.

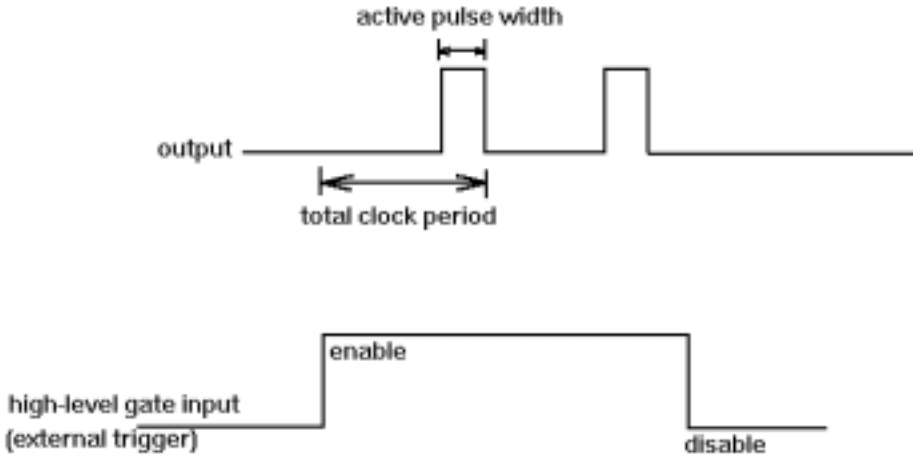

Figure 103: A Signal Generated by the Counter/Timer

Note: Some gate types may not be supported by your hardware. Refer to the supported options table in your hardware device driver documentation.

Delay and Pulse Width

Delay and *Pulse Width* are an alternative to using *Frequency* and *Duty Cycle* to define the shape of the counter/timer output, and are typically used with one-shot mode. Delay is the amount of time in milliseconds (ms) between the beginning of the counter/timer output and the active portion of the pulse. The pulse width is the duration of the active portion of the pulse in milliseconds, as shown in Figure 104.

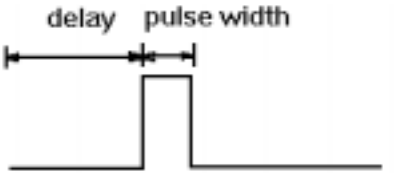

Figure 104: Delay and Pulse Width of a Counter/Timer's One-Shot Signal

Duration

When the counter / timer is set to measure the frequency of events (pulses in a TTL signal), the *Duration* is the total duration over which the signal is measured, in seconds, as shown in Figure 105.

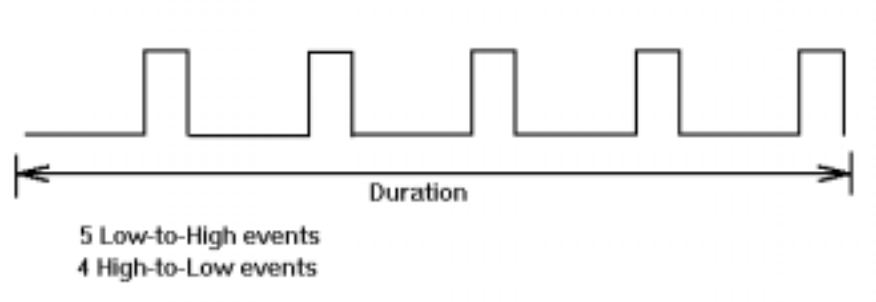

Figure 105: Measuring the Frequency of Events

Clock Frequency

If an internal clock source is selected, then its frequency is determined by entering a value in Frequency (Hz) in the C / T Configuration dialog box. If an external clock source is selected, then enable DT VPI to be aware of its frequency by entering its frequency (or the closest you can estimate) in the External Clock Frequency (Hz) edit box.

Using the Counter/Timer as a Clock

The counter / timer can be used to generate a *user-supplied signal* for use as the pacer clock for a data acquisition instead of the onboard oscillator. The counter / timer is configured to output a generated signal of a certain frequency; this signal is then used by the A / D subsystem as an external clock. To accomplish this, the output from the counter / timer (for example, C / T OUT1 on the screw terminal panel) must be connected to the A / D external clock input (A / D EXT CLK on the screw terminal panel). Figure 106 shows an example of using the counter / timer as a clock.

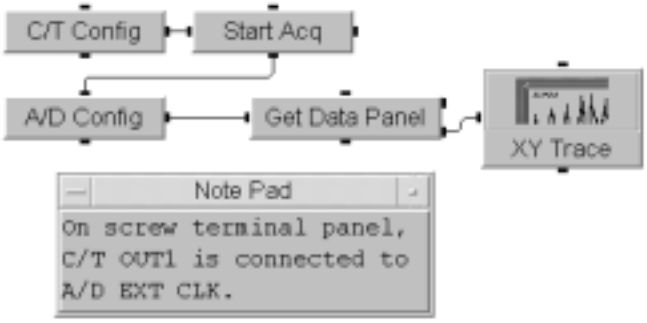

Figure 106: Using a Signal Generated by the C/T as the External Clock to the A/D Subsystem

Note: When it executes, C/T Config object initializes the counter / timers on any Data Translation data acquisition board. The initialization causes a spike on the output of the counter / timer. Therefore, if you are using the counter / timer as an external clock or external trigger, ensure that the Start Acq object is executed for the counter / timer subsystem before starting the A / D subsystem.

Configuring the C/T as a Clock

To use the counter/timer circuit as a clock, select **DT DataAcq ⇨ C/T Config**, and click **Configure**. Set the mode to **Generate Rate**. If you are using the internal clock as the source, select Internal under Clock Source and enter in the desired frequency of the clock in the Frequency (Hz) edit box. If you are using an external frequency as the clock source, select External and enter the estimated frequency in the External Clock Frequency (Hz) edit box.

Indicate the pulse type (High-to-Low or Low-to-High) and its width in the Duty Cycle (%) edit box.

If you select a gate source other than None, the square wave output is generated based on the gate specified (High Level, Low Level, Rising Edge, Falling Edge).

Select **DT DataAcq ⇨ A/D Config**. Now, click the **Configure** button on the A/D Config object. (If you have more than one data acquisition board in your computer, then select the A/D subsystem you want to interact with in the Select Subsystem dialog box.) In the A/D Configuration dialog box, click the **Timing** button. Set the Clock Source to External.

After connecting a signal source to channel 0 (and you have already connected the output from the first C/T to the A/D external clock input on the screw terminal panel), the program is ready to run.

What's Really Going On?

When you run this program, the Start Acq object executes and starts the counter / timer subsystem. The counter / timer begins generating a signal at the specified frequency, and outputs it. That signal reaches the A / D subsystem's external clock input (because they are connected on the screw terminal panel), and analog data is acquired using the external clock instead of the onboard oscillator. Figure 107 shows the timing diagram for a counter / timer as a clock.

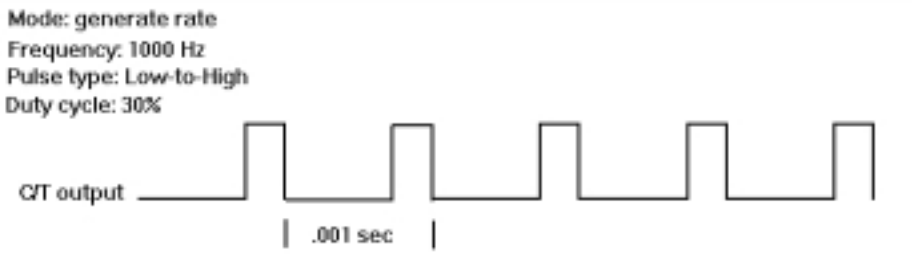

Figure 107: Timing Diagram for a Counter/Timer as a Clock for Acquisition

Using the Counter/Timer as a Trigger

The counter / timer can be used as a trigger to start analog acquisition (or analog output). The counter / timer subsystem is configured to output a single pulse (called a *one-shot*); this pulse is then recognized by the A / D subsystem as the external trigger to start acquiring data. To accomplish this, the output from the counter / timer (for example, C / T OUT1 on the screw terminal panel) must be connected to the A / D external trigger input (A / D EXT TRIG on the screw terminal panel).

Take a look at the example program CTRIGGER.VEE (in the \Examples\Dataacq\ directory), shown in Figure 108.

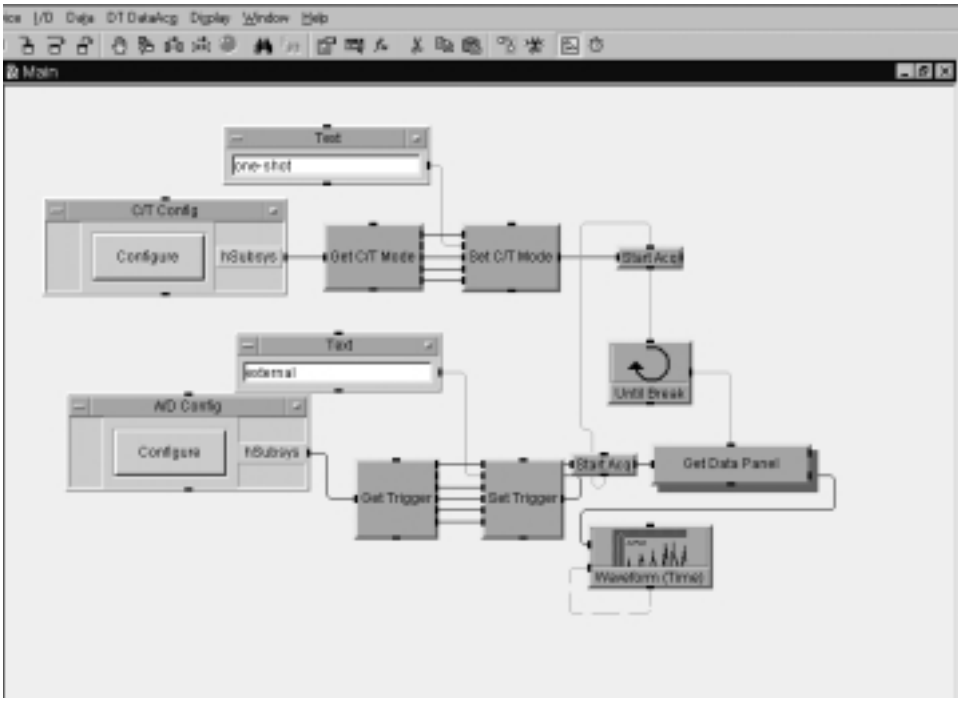

Figure 108: Example on Disk: CTRIGGER.VEE

Note: When it executes, the C/T Config object initializes the counter/timers on any Data Translation data acquisition board. The initialization causes a spike on the output of the counter/timer. Therefore, if you are using the counter/timer as an external clock or external trigger, ensure that the Start Acq object is executed for the counter/timer subsystem before starting the A/D subsystem.

Configuring the C/T as a Trigger

To configure this example program, click **Configure** on the C/T Config object. If you have more than one counter/timer subsystem on your board, select the subsystem whose output you connected to the A/D external trigger input on the screw terminal panel. Click **OK**. Then, set the Mode to One-Shot, leaving the Delay at 0.5 ms and the Pulse Width at 0.5 ms. Click **OK** to close the C/T Configuration dialog box.

Now, click the **Configure** button on the A/D Config object. If you have more than one data acquisition board in your computer, select the A/D subsystem you want to use in the Select Subsystem dialog box. In the A/D Configuration dialog box, click the **Timing** button. Set the Trigger Source to External.

After connecting a signal source to channel 0 (and you have already connected the output from the first C/T to the A/D external trigger input on the screw terminal panel), the example program is ready to run.

What's Really Going On?

When you run this program, the Start Acq object is executed, which starts the delay period for the one-shot pulse. (The Get C/T Mode and Set C/T Mode objects ensure that the counter / timer is set for one-shot mode, even if you did not configure it as such. Similarly, the Get Trigger and Set Trigger objects ensure that the trigger is external.) When the delay period (0.5 ms) is over, a pulse of 0.5 ms is output from the board and reaches the A / D external trigger input. The A / D subsystem, which has been waiting for the external trigger, then begins acquiring data. Figure 109 shows the timing diagram for this example.

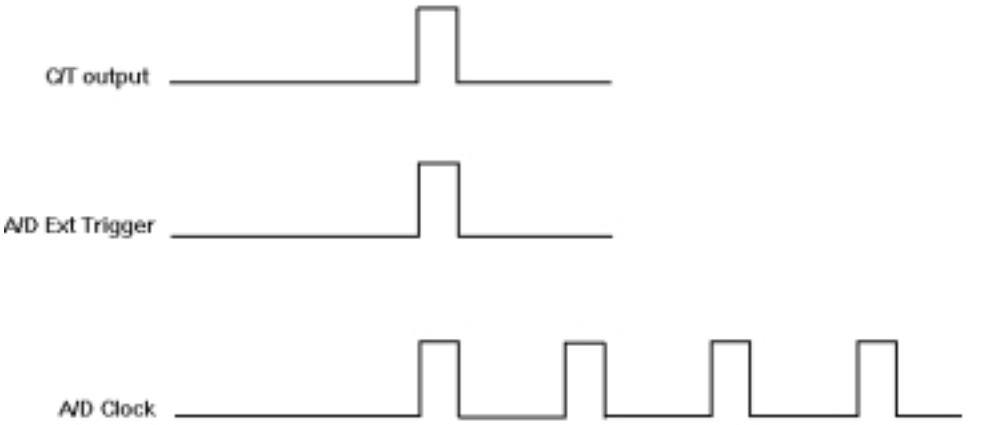

Figure 109: Timing Diagram for a Counter/Timer used as a Trigger To Start Acquisition

Taking it Further: An External Trigger to the C/T

Using the counter/timer subsystem as an external trigger to the A/D or D/A subsystem gives you flexibility in determining when the A/D or D/A starts. Although the value for Delay in the C/T Configuration dialog box cannot be set for much more than 0.5 s, you can control programmatically when the Start Acq object is executed and, therefore, control when the trigger occurs.

Furthermore, if you want to externally trigger the A/D or D/A subsystem but need to use a different type of signal than the one allowed in the A/D Configuration dialog box under Timing, then supply an external trigger to the C/T subsystem itself! To do this, set the C/T Gate to an External source, and then choose from among the choices of Gate Types. Figure 110 shows the timing diagram for this example.

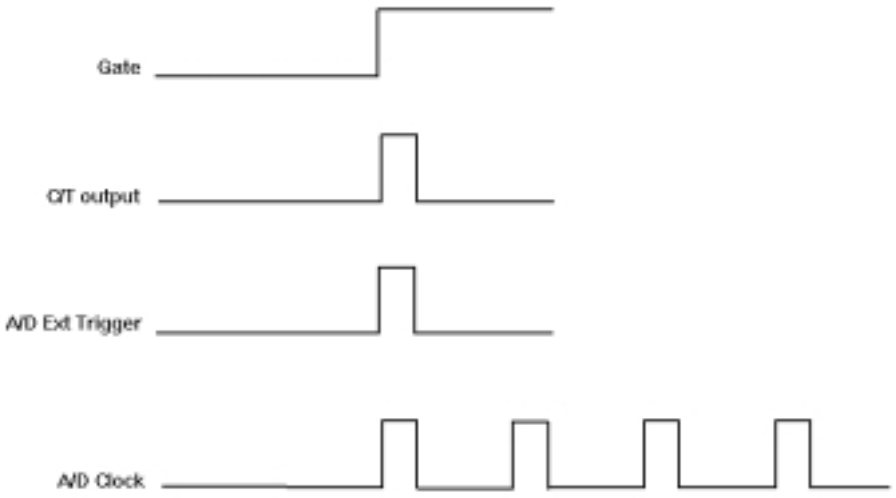

Figure 110: Timing Diagram for a Counter/Timer used as a Trigger To Start Acquisition

Using the Counter/Timer to Count Events

The counter/timer counts the number of events on the internal or external source input. In most cases, you are interested in counting external events, as opposed to the board's internal oscillator pulses. To count the number of events in an external signal input, connect an external signal to SRC1 on the screw terminal panel, then connect the ground lead. A program using the C/T subsystem to count events is shown in Figure 111.

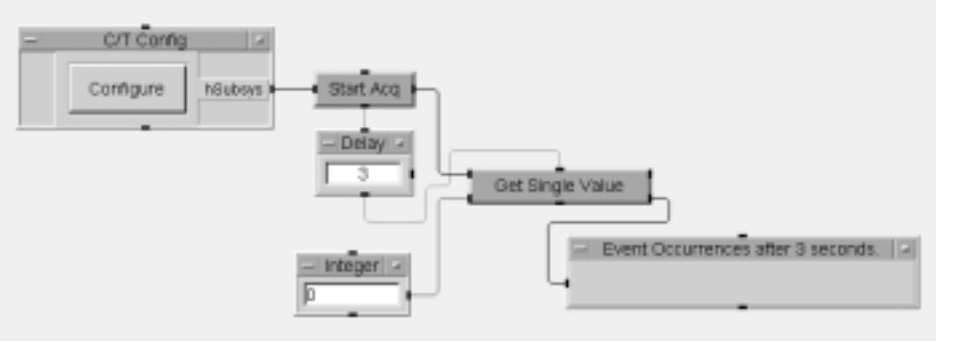

Figure 111: A Program Using the Counter/Timer To Count Events in an External Input Signal

Configuring the C/T to Count Events

Select the C/T Config object and click **Configure**. Set the mode to Count Events. See your hardware user manual for more information on the definition of an event.

Under Clock Source, select Internal or External. Typically, you set the gate source to External events are counted based on the gate specified (High Level, Low Level, Rising Edge, Falling Edge). For this example, the clock source is an external signal of type High Level. The appropriate gate input and ground connections were made.

What's Really Going On?

When you run this program, the C/T subsystem begins counting the number of events on the clock source input. The Get Single Value object reads the total number of events and the current tally appears on the AlphaNumeric display object. The counting continues until you stop the subsystem with a Stop Acq object, or reconfigure the subsystem. Figure 112 shows the timing diagram for this example.

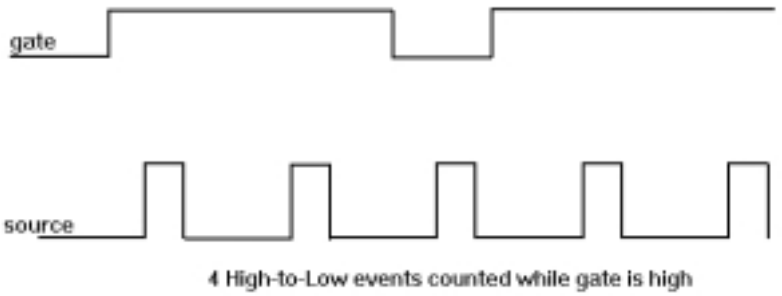

Figure 112: Timing Diagram for a C/T UsedTo Count Events

Using the Counter/Timer to Measure Frequency

To measure the frequency of a TTL signal, the counter/timer counts the number of events on the internal or external source input over a specified time. In most cases, you are interested in measuring the frequency of occurrence of external events, as opposed to the board's internal oscillator pulses. To measure the frequency of events in an external signal input, connect an external signal to SRC1 on the screw terminal panel; then connect the ground lead. Just for fun, measure the frequency of your data acquisition board's internal oscillator.

A program using the C/T subsystem to measure the frequency of the internal oscillator looks like that shown in Figure 113. Note that the Get Data Panel object acquires only one point (the frequency) on channel 0.

Figure 113: A Program Using the Counter/Timer To Measure the Frequency of the Board's Internal Clock

Configuring the C/T to Measure Frequency

Select the C/T Config object, and click **Configure**. Set the mode to Measure Frequency. See your hardware user manual for more information on the definition of an event.

Under Clock Source, select **Internal** or **External** depending on whether you want to measure the frequency of the internal clock or an external signal. For this example, the clock source is the internal oscillator, though measuring your internal oscillator is not very useful. (For this example, the pulse type can be either High-to-Low or Low-to-High.) If you set the Gate Source to External, then frequency of occurrence is measured only when the specified gate (High Level, Low Level, Rising Edge, Falling Edge) is active.

Under Duration (secs), indicate the period over which you want to count events. In terms of accuracy, the more periodic events you count, the more accurate the result. If you count ten periodic events, the accuracy is within ±10%. If you count 100 period events, the accuracy is within ±1%.

$$\left(\frac{1}{frequency} \bullet \frac{1}{duration}\right)100 = PercentAccuracy$$

What's Really Going On?

When you run this program, the counter / timer begins counting events on the internal or externally supplied clock. In this example, we are measuring the frequency of the internal oscillator. When the Get Data Panel object is executed, the counter / timer subsystem starts automatically, and counting begins. The counting lasts for the time specified in the Duration (secs) edit box. The result is output from the Get Data Panel object when the specified duration elapses. The result is in the form of a one-dimensional array of size 1. Figure 114 shows the timing diagram for this example.

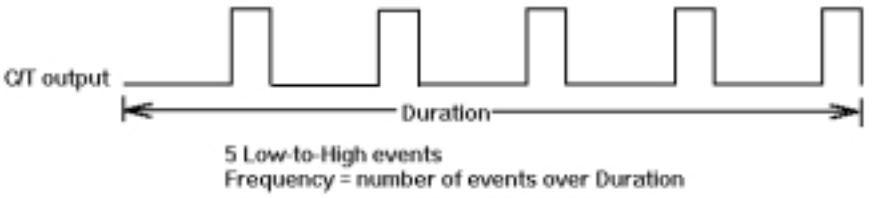

Figure 114: Timing Diagram for a C/T Used To Measure the Frequency of Occurrence

Optimizing Counter/Timers

When you set the counter/timer subsystem to output a one-shot pulse, you specify certain values for the delay and pulse width. The data acquisition hardware accommodates your settings as closely as possible. The following algorithm is used by the hardware to calculate the actual delay and pulse width:

$$T = \frac{1}{frequency} = period$$

Set the parameters as follows:

1. Set $T = Delay + PulseWidth$.

2. Read the actual value of T from the board.

3. Set $DutyCycle = 1 - \frac{Delay}{actualT}$.

4. Read the actual value of *DutyCycle* from the board.

5. Set $PulseWidth = actualDutyCycle \bullet actualT$.

6. Set $Delay = actualT - PulseWidth$.

Developing Programs for VEE RunTime

This chapter provides some tips for using VEE and DT VPI to develop data acquisition programs to run in the VEE RunTime environment. It is written for those users who want to develop a program in VEE:

- *Make all modifications to your program and debug it in the VEE environment.* Once you have transferred the program to the VEE RunTime environment, you cannot change it.
- *Develop the program with a panel view designed for the appropriate window size.* You may be running VEE on a computer with an Ultra-VGA monitor. You can use the full capabilities of that monitor in developing the detail view. However, the panel view must be developed to fit on the display that your run-only systems use. For example, if your run-time systems use 640-by-480 pixel VGA monitors, the panel view must fit within that space.

Note: If it has no panel view, your program still runs under VEE RunTime, but with a blank screen. (However, Show Panel on Execute pop-up UserObjects still appears.)

- *Leave any data acquisition subsystem configuration objects unconfigured.* If the system running VEE RunTime has a different data acquisition board or configuration, then an error occurs prior to allowing the operator to select a new board. To avoid this error, leave the subsystem configuration objects unconfigured in the VEE program, and require the operator to configure each subsystem configuration object.

The subsystem configuration objects are A/D Config, D/A Config, Digital In Config, Digital Out Config, and C/T Config. These objects are in an unconfigured state immediately after they are taken off the DT DataAcq menu, before you click the Configure button on the object.

- *If your program is intended to run with a variety of data acquisition boards, design your program to check for the user's system capabilities.* For example, if your program needs an external trigger, then check for the availability of an external trigger with the Get Trigger object (**DataAcq ⇨ Get Config ⇨ Get Trigger**). If an external trigger is not available, your program should alert the operator.
- *Provide a means of stopping the program from the panel view.* If you start a program in the RunTime environment with the -r option, no tool bar is available; therefore, no **Stop** button is available. The operator can stop the program by pressing Ctrl+C, but it is good practice to provide a button in the panel view to stop the program.
- *If your I/O is not embedded in your program, transport your VEE.IO file to the VEE RunTime environment.* When you transport your program to the run-only environment, include your VEE.IO file. Put the file in the C:\PROGRAM FILES\AGILENT\VEE PRO RUNTIME 6.0 directory of each run-only system that will run your program. If VEE.IO is not in that directory, a default VEE.IO file is created and your program may not run correctly.
- *Load any data acquisition device drivers used by your program.* Use the Windows Control Panel icon, according to the user manual provided with your hardware's device driver.

- *In some cases, you may want to secure your program.* You do not have to secure your program since VEE RunTime shows only the panel view and does not let the operator save the program or modify its functionality. However, it is possible for the operator to alter what is seen on the screen.

  For example, the operator can move objects, change their size, or remove them from the panel view when the program is not running. This has no effect on how the program functions, and the changes cannot be saved. However, you can prevent this entirely if you secure your program before transferring it to the run-only environment.

Note: In VEE, don't secure your only copy—you cannot "unsecure" a program once it is secured and saved.

- *Document your program.* VEE provides the Save Documentation menu option to extract information from a program.

The best source of VEE training for operators is the tutorial, which you can access from the help menu. In addition, you may want to write a short manual for operators of your VEE program.

Here are some other things to consider when developing your program:

- If it has no panel view, your program still runs in VEE RunTime, but with a blank screen.
- You cannot **Step** your program in VEE RunTime.
- Breakpoints saved with your program are ignored when running under VEE RunTime.

Advanced Functionality

Calling the Advanced Functions 229
Accessing the Unsecured Data Acquisition UserObjects 240
Thermocouple Support 241

DT VPI provides additional data acquisition functionality that cannot be found in the **DT DataAcq** menu. You can access DT VPI's advanced features in the following ways:

- By calling advanced functions in a Call Function object;
- By using or modifying the unsecured versions of the secured data acquisition UserObjects; and
- By merging the thermocouple UserFunctions, and using or modifying them in your own programs.

Each method is discussed in the following sections.

Calling the Advanced Functions

Advanced functionality is accessed using a Call Function object (**Device ⇨ Function ⇨ Call**). The Call Function object's object menu ⇨ **Select Function** command enables you to select from among the advanced data acquisition functions.

Note: You must configure a subsystem object before attempting to select an advanced data acquisition function in a Call Function object. Alternatively, use an Import Library object to import the library of compiled functions called DTVEE.DLL, whose definition file is DTVEE.VH.

The following subsections describe the advanced data acquisition functions.

dtv.ConfigSubSys

dtv.ConfigSubSys enables selection, opening, and configuration of a subsystem via dialog boxes. If ConfigInfo is valid, then dtv.ConfigSubSys opens the subsystem indicated by ConfigInfo, and loads the stored configuration. A call to dtv.ConfigSubSys is the initial call to open any subsystem for use with other objects.

Table 10: dtv.ConfigSubSys

Terminal	Input or Output	Type	Description
Ret Value	output	Int32	subsystem handle
EditConfig	input	Int32	description
ConfigInfo	input and output	Int32 Array	description

dtv.GetChannelType

dtv.GetChannelType gets the current channel type. The channel type can be either single-ended or differential.

Table 11: dtv.GetChannelType

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
ChannelType	output	Enum	singleended or differential

dtv.SetChannelType

dtv.SetChannelType sets the channel type to single-ended or differential.

Table 12: dtv.SetChannelType

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
ChannelType	output	Enum	singleended or differential

dtv.GetCriticalThreshold

dtv.GetCriticalThreshold gets the current critical threshold value, in seconds, to Threshold. The critical threshold is the minimum amount of data allowed in the Mixed Data Buffer before a channel goes to the Done or Waiting state (depending on the data mode).

Table 13: dtv.GetCriticalThreshold

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Threshold	output	Real	the current threshold value in seconds

dtv.SetCriticalThreshold

dtv.SetCriticalThreshold sets the critical threshold value, in seconds, to Threshold. The critical threshold is the minimum amount of data allowed in the Mixed Data Buffer before a channel goes to the Done or Waiting state (depending on the data mode).

Table 14: dtv.SetCriticalThreshold

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Threshold	input	Real	the new threshold value in seconds

dtv.GetData

dtv.GetData retrieves the specified number of samples from the input subsystem pointed to by hSubsys. dtv.GetData does not loop while waiting for the samples to be available, as do the Get Data Panel and Get Data Function objects. If the requested number of points is not available, dtv.SetData returns a scalar in Data indicating the number of points in the channel's queue. If the subsystem is not running and autostart is true, dtv.GetData starts the subsystem. If the subsystem is stopped, dtv.GetData posts an error.

Table 15: dtv.GetData

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Channel	input	Int32	channel number
Points	input	Int32	number of samples to acquire
Data	output	Real Array	acquired data
BufferStart	output	Real	time duration measured from time of acquisition start to time of arrival of first sample in each buffer. Useful when using Allow Gaps mode to inform you of how much acquisition time was spent on a gap of data.

dtv.GetPreTrigData

dtv.GetPreTrigData retrieves the specified number of samples from the input subsystem pointed to by hSubsys. dtv.GetPreTrigData does not loop while waiting for the samples to be available, as do the Get Pre-Trigger Data Panel and Get Pre-Trigger Data Function objects. If the requested number of points is not available, dtv.GetPreTrigData returns a scalar in Data indicating the number of points in the channel's queue. If the subsystem is not running and autostart is true, dtv.GetPreTrigData starts the subsystem. If the subsystem is stopped, dtv.GetPreTrigData posts an error.

Table 16: dtv.GetData

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Channel	input	Int32	channel number
Points	input	Int32	number of samples to acquire
Data	output	Real Array	acquired data
BufferStart	output	Real	time duration measured from time of acquisition start to time of arrival of first sample in each buffer. Useful when using Allow Gaps mode to inform you of how much acquisition time was spent on a gap of data.

dtv.PutData

dtv.PutData puts the samples in Data to the output subsystem pointed to by hSubsys on the specified channel. If the subsystem is not running and autostart is true, dtv.PutData starts the subsystem. dtv.PutData posts an error if the subsystem is stopped or if the data queue is full and the subsystem is not yet running. PutDone contains non-zero if enough room was available on the data queue for the samples, and contains zero if not enough room was available on the data queue.

Table 17: dtv.PutData

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Channel	input	Int32	channel number
Data	input	Real Array	data to output
PutDone	output	Int32	status

dtv.GetEncoding

dtv.GetEncoding returns the type of data encoding used when converting from analog to digital or digital to analog.

Table 18: dtv.GetEncoding

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Resolution	output	Int32	encoding: 0=binary, non-0= twos complement

dtv.GetFileSize

dtv.GetFileSize returns the size of any specified file.

Table 19: dtv.GetFileSize

Terminal	Input or Output	Type	Description
FileName	input	text	pathname to file
File Size	output	Int32	file size (bytes)

dtv.GetMixedBufferSize

dtv.GetMixedBufferSize retrieves the size (in seconds) of the internal mixed data buffer.

Table 20: dtv.GetMixedBufferSize

Terminal	Input or Output	Type	Description
hSubSys	input	Int32	subsystem handle
hSubSys	output	Int32	subsystem handle
MBSize	output	Real	size in seconds of the mixed data buffer

dtv.SetMixedBufferSize

dtv.SetMixedBufferSize sets the size (in seconds) of the internal mixed data buffer.

Table 21: dtv.SetMixedBufferSize

Terminal	Input or Output	Type	Description
MBSize	input	real	size in seconds of the mixed data buffer (min: 0.5 seconds; max: 10 seconds)
hSubSys	output	Int32	subsystem handle

dtv.GetResolution

dtv.GetResolution returns the resolution of the subsystem pointed to by hSubsys.

Table 22: dtv.GetResolution

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Resolution	output	Int32	number of bits of resolution

dtv.SetResolution

dtv.SetResolution sets the resolution of the subsystem pointed to by hSubsys.

Table 23: dtv.SetResolution

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Resolution	input	Int32	number of bits of resolution

dtv.GetSingleValueEx

dtv.GetSingleValueEx allows you to retrieve a single value from a subsystem on a specified channel, using a specified gain.

Table 24: dtv.GetSingleValueEx

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Channel	input	Int32	subsystem channel #
Gain	input	Real	Gain
hSubSys	output	Int32	subsystem handle
Value	output	Int32	single raw data value

dtv.PutSingleValueEx

dtv.PutSingleValueEx allows you to output a single value on a specified channel, using a specified gain.

Table 25: dtv.PutSingleValueEx

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Channel	input	Int32	subsystem channel #
Gain	input	Real	Gain
hSubSys	output	Int32	subsystem handle
Value	output	Int32	single raw data value

dtv.GetTriggerThreshold

dtv.GetTriggerThreshold returns the current trigger threshold level of the subsystem pointed to by hSubsys.

Table 26: dtv.GetTriggerThreshold

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Threshold	output	Int32	trigger threshold level

dtv.SetTriggerThreshold

dtv.SetTriggerThreshold sets the trigger threshold level in DAC0. When the analog input signal passes the threshold level, the subsystem begins to report the data samples. Boards, such as the DT2839 and DT3010 Series, that support threshold triggering can use this function. Typically, the input to hSubsys is the subsystem handle for the A/D subsystem.

Table 27: dtv.SetTriggerThreshold

Terminal	Input or Output	Type	Description
hSubsys	input	Int32	subsystem handle
Ret Value	output	Int32	subsystem handle
Threshold	input	Int32	trigger threshold level

Accessing the Unsecured Data Acquisition UserObjects

If you need to tinker with the inner workings of the subsystem configuration objects or the get and put data objects, you have access to the unsecured versions of those UserObjects. Select **File ⇨ Merge** or **File ⇨ Open** and select one of the following files from the \VEE Pro 6.0\DataAcq or \VEE OneLab 6.0\DataAcq directory:

Table 28: Files Containing the Unsecured Versions of the UserObjects

Filename	Description
adconfig.vee	A/D Config object
ctconfig.vee	C/T Config object
daconfig.vee	D/A Config object
diconfig.vee	Digital In Config object
doconfig.vee	Digital Out Config object
getdataf.vee	Get Data Function object
getdatap.vee	Get Data Panel object
getptdataf.vee	Get Pre-Trigger Data Function object
getptdatap.vee	Get Pre-Trigger Data Panel object
putdataf.vee	Put Data Function object
putdatap.vee	Put Data Panel object

Thermocouple Support

In addition to advanced function calls, DT VPI provides thermocouple support in the form of three UserFunctions. These objects can be merged with your own programs; you can then use and modify them as needed.

To merge them into your program, select File ⇨ Merge and search for the file \LIB\THRMOCPL.VEE. Alternatively, select File ⇨ Merge Library, search for the file \LIB\THRMOCPL.VEE, and then use Call Function objects to call the functions ThAverage, CompensationVolts, and VoltstoDegrees.

Example on Disk: The example program MONITOR.VEE (in the \Examples\DataAcq\ directory) demonstrates the use of the A/D subsystem to collect and display current thermocouple temperatures.

Example on Disk: The example program TEMP727.VEE (in the \Examples\DataAcq\ directory) demonstrates the use of the DT727-T expander board to acquire thermocouple data. Using the selected thermocouple channel, the program determines which expander channel has the cold-junction compensation circuit routed to it. It then acquires and calculates the compensation voltage to use in all remaining temperature acquisitions.

ThAverage

Sorts the input array, discarding the highest value and the lowest value. Returns the average of the remaining values.

Table 29: ThAverage

Terminal	Input or Output	Type	Description
X	input	Any	Array
Result	output	Any	Average of values in array

CompensationVolts

Computes the compensation voltage for a given temperature for a given thermocouple type.

Table 30: CompensationVolts

Terminal	Input or Output	Type	Description
Temp	input	Real	Temperature in degrees Celsius
Thermocouple	input	Text	Thermocouple Type: must be B, E, J, K, R, S, or T
Result	output	Real	Compensation voltage

VoltsToDegrees

Computes the temperature in degrees Celsius for given voltage and compensation voltage values.

Table 31: VoltsToDegrees

Terminal	Input or Output	Type	Description
Voltage	input	Real	Voltage value.
Compensation Voltage	input	Real	Value representing the cold junction compensation voltage. This value is added to Voltage prior to conversion.
Result	output	Real	Temperature in degrees Celsius.

Example Programs

The following is a summary of the data acquisition example programs. These programs are located in the directory \Program Files\Agilent\VEE Pro 6.0\Examples\Dataacq\ or \Program Files\Agilent\VEE OneLab 6.0\Examples\Dataacq\.

- **727** demonstrates how to acquire data using the DT727 Channel Expansion Panel.
- **ADCOLUMN** illustrates how to save three channels of A/D data as columns to import to programs such as Microsoft Excel.
- **ALLCHAN** illustrates how to create a multi-channel data file using the Get and Set Multi Channel Mode objects as well as using the dtv.GetFileSize function. Use this example with multfile.
- **ADSIMPLE** acquires and displays 512 values from the A/D subsystem.
- **ADBURST** acquires and displays a burst of 512 samples from the A/D subsystem.
- **ADSINGLE** acquires a user-defined number of values one at a time, then builds and displays the resulting waveform.
- **ATLABFIL** reads data stored by Data Translation's ATLAB library.
- **CAL2801** illustrates calibration of a DT2801 Series board.
- **CAL2812** illustrates calibration of a DT2812 Series board.
- **CAL2821** illustrates calibration of a DT2821 Series board.
- **CALCJC** illustrates calibration of the cold junction compensation circuit for a DT707-T, DT707A-T, or a DT717-T.
- **CNTEVENT** counts events using the counter/timer subsystem.
- **COHERENCE** calculates the coherence for two incoming waveforms.

- **CONFFT**demonstrates continuous acquisition and FFT processing.
- **CTRIGGER** uses the counter/timer subsystem to generate a trigger for the A/D subsystem.
- **DABURST** outputs a burst of 256 samples to the D/A subsystem.
- **DACONT** continuously outputs data to the D/A subsystem.
- **DAFILE** creates a file and outputs it to the D/A subsystem.
- **DASIMPLE** outputs 256 samples to the D/A subsystem.
- **DASINGLE** outputs a user-specified value using the D/A subsystem.
- **DDE** illustrates DDE to Excel.
- **DIN** illustrates continuous digital input.
- **DOUT** illustrates continuous digital output.
- **DIO** demonstrates reading from a digital input subsystem and writing to a digital output subsystem.
- **DTVFILE** reads and displays data from a file containing previously acquired data. Illustrates how to parse DT VPI's data acquisition file format.
- **EASYPK** is a simple peak detection example.
- **FILENAME** acquires data to a disk file. A new file is used for every 512 samples.
- **HICHAN1** demonstrates collecting samples using high channel counts.
- **HICHAN2** demonstrates the use of Get and Set Configuration objects to programmatically change the channel list.
- **MAGBIN** illustrates how to find the magnitude of a selected bin from FFT results.

- **MEASURE** measures a frequency using the counter/timer subsystem.
- **MONITOR** acquires and displays current thermocouple temperatures using an A/D subsystem and a screw terminal panel with software cold junction compensation.
- **MULTFILE** illustrates how to read the DT VPI multi-channel data file and display results. Use this example with allchans.
- **MULTI** acquires data continuously on four channels.
- **PCLABFIL** reads data from a file written by Data Translation's PCLAB library.
- **PRETRIG** demonstrates how to acquire pre-trigger data.
- **PSD** illustrates a power spectral density with averaging, using feedback to accumulate the average FFT results.
- **SERIALIO** demonstrates serial communication using the computer's COM2 and a dumb terminal.
- **SIMULTAN** simultaneously inputs and outputs data using the A/D and D/A subsystems.
- **SLOWAD** demonstrates how to alter the channel buffer size for slow sampling rates.
- **TEMP727** demonstrates the use of the DT727-T expander board to acquire thermocouple data.
- **TOFILE** demonstrates how to monitor the progress of file output from an A/D subsystem.
- **THRESH** demonstrates how to implement voltage threshold triggering for an A/D subsystem.
- **TRANSAVG** illustrates how to implement a transfer function with averaging.

- **TRIGGER** acquires data using a user-specified external or internal trigger.
- **TRIGSCAN** acquires data using triggered scan mode.

Error Codes

DT VPI Errors 250
DT-Open Layers Errors 258

DT VPI Errors

This appendix contains reference information on the errors that are raised by data acquisition objects. For a description of VEE errors, refer to your VEE documentation.

Error Number	Description
0	No error
2000	DT Open Layers error: <descriptive error string>. Refer to the DT-Open Layers errors listed at the end of this appendix.
2001	Invalid device number passed to routine. The hSubsys is invalid.
2002	Unsupported device type. The requested parameter cannot be set because it is not supported by the subsystem.
2003	Fatal internal DLL error. Please note what you were doing and call Data Translation. This error is fatal, you should exit and restart Windows.
2005	Invalid subsystem type passed to a config object. This error is raised if an invalid subsystem type is passed to an advanced configuration object.
2006	Trigger Error: Acquisition Stopped. The subsystem received an external trigger while running.
2007	Underrun error. D/A Operation clocked too fast. A hardware error; the hardware cannot keep up with the clock rate. Slow down the output rate.
2008	Overrun error. A/D Operation clocked too fast. A hardware error; the hardware cannot keep up with the clock rate. Slow down the acquisition rate.
2009	Event error. A hardware error; the counter/timer subsystem in Count Events mode cannot keep up with the incoming events. Slow down the event rate.

2010 Operation cannot be performed while the system is running. Use the Acq Status object to check the status of the subsystem to avoid this error.

2012 The Acquisition or Output is complete for the channel. This is typically caused when processing cannot keep up with the acquisition or output rate. The A/D or D/A subsystem has stopped. This error is raised when you try to read or write samples to a device that has stopped.

For the A/D subsystem, the subsystem stops when all of the memory set aside for incoming data is used. To prevent this error, you can do one or all of the following:

- Enable burst mode if you know the number of samples required or the duration over which you want to acquire.
- Enable Allow Gaps data mode.

For the D/A subsystem, the subsystem stops when no more data is available in memory for the D/A subsystem to output. To prevent this error, you can do one or all of the following:

- Enable burst mode if you know the number of samples to output or the duration over which you want to output.
- Enable Allow Gaps data mode.
- Slow the output rate.
- Reduce processing of the outgoing data in VEE.

2013 Could not create message processing window. VEE could not load necessary libraries. Restart Windows.

2014 The specified subsystem has been allocated by another process. The Windows driver for the subsystem is already being used by another application.

2015 There is insufficient free memory to complete the requested function. VEE needs more memory. Close down other applications to free memory.

2017	There are no boards installed with the specified subsystem type. The installed board does not have the subsystem you are trying to configure.
2018	An internal resource error has occurred. VEE had an error obtaining Windows resources. Save your program and reboot.
2019	Internal DLL error. Please note what you were doing and call Data Translation.
2020	The user pressed cancel during required configuration. If multiple subsystems are installed, one must be selected to execute a subsystem configuration object.
2021	The Channel Number entered is invalid for the current operation. Either the channel is not on the current channel list, or it is a channel that is not supported by the subsystem.
2022	More data points than can be accepted by the current Channel Buffer Size have been put to this channel. A Put Data Panel or Put Data Function object tried to write too many data points to a D/A subsystem channel that has not been started.
2023	Too many data points were requested from a subsystem. A Get Data Panel or Get Data Function object was executed with a C/T subsystem hSubsys and the number of points specified was greater than one.
2024	Input must be Int32.
2025	Input must be Real.
2026	Input must be Text.
2027	Input must be Enum.
2028	Input arrays must be of the same size and shape.

2029	The saved device configuration is invalid. Please reconfigure the device. The program was saved with a configuration that included a board name and subsystem type that does not match the present configuration.
2030	Can not find any supported hardware. No DT-Open Layers Windows driver and board was found in the system.
2031	Analog output device required for threshold triggering in use. Please close the analog output device on this board. The analog threshold trigger requires the D/A subsystem to be closed.
2032	Requested threshold trigger voltage is out of range. The threshold trigger voltage must be in the range specified in the A/D Hardware Dialog.
2033	Duplicate filename <file name> in list. Two channels in the Channel list have the same file name. Change one of file names to a unique name.
2034	The filename <file name> is already used. The filename added to the list is used for another channel, use a unique filename.
2035	The file <file name> already exists. The file already exists; use a unique file name.
2036	Bad file name or directory: <file name>. Most likely the filename or directory does not exist.
2037	Write protect error on file <file name>. The file is write protected.
2038	Test file can not be deleted. Create-no-modify network share points are not supported. The file cannot be written to a disk drive with no delete privileges.
2039	The file <file name> does not exist. The file specified for the D/A subsystem to output cannot be found. Check the name and path.

2040	Error writing to file <file name>. Check your drive. The disk may be full or there may not be a disk in the floppy drive.
2041	Error re-opening file <file name>. A system resource problem was encountered. Windows may be out of file handles.
2042	Error reading from file <file name>. The file could be corrupted or deleted.
2043	File <file name> has a bad header. The file header is not a data acquisition file header. See Appendix C for more information.
2044	Data width in file <file name> does not match device. The data width specified by the file header is not the same as the data width required by the subsystem.
2045	Resolution in file <file name> does not match device. The resolution specified by the file header is not the same as the resolution required by the subsystem.
2046	Data encoding in file <file name> does not match device. The data encoding in the file header is not the same as the data encoding required by the subsystem.
2047	Voltage range or gain in file <file name> does not match device. The voltage range in the file header as determined from minvolts and maxvolts is not the same as the range required by the subsystem.
2048	System error opening file <file name>. Generic error, could be a non-DOS disk, a bad file sector, or access denied for example.
2049	The file <file name> is already open. The file is open and cannot be shared. Close the file.
2050	File <file name> has an illegal size. Partial samples are not allowed.

2051	Requested Rate exceeds the maximum throughput of the Counter/Timer device. Choose a slower rate. Consult your data acquisition hardware manual for maximum rate information.
2052	Requested Rate is less than the minimum throughput of the Counter/Timer device. Choose a faster rate. Consult your data acquisition hardware manual for minimum rate information.
2053	Requested Rate exceeds the maximum throughput available with the current external clock frequency. Check your external clock rate and adjust the information given to DT VPI.
2054	Requested Rate is less than the minimum throughput available with the current external clock frequency. Check your external clock rate and adjust the information given to DT VPI.
2055	Combination of Delay and Pulse Width exceeds the maximum throughput of the Counter/Timer device. 1/(Delay + Pulse Width) must be less than the maximum frequency of the clock source.
2056	Combination of Delay and Pulse Width is slower than the minimum throughput of the Counter/Timer device. 1/(Delay + Pulse Width) must be greater than the minimum frequency of the clock source.
2057	Combination of Delay and Pulse Width exceeds the maximum throughput available with the current external clock frequency. 1/ (Delay + Pulse Width) must be less than the maximum frequency of the clock source.
2058	Combination of Delay and Pulse Width is slower than the minimum throughput available with the current external clock frequency. 1/(Delay + Pulse Width) must be greater than the minimum frequency of the clock source.

2059	Requested Measure Frequency Duration is out of range. The frequency duration must be between 1 ms and 4,294,967,296 ms.
2060	Invalid Counter/Timer mode for this operation. Parameters set with a Set Config object are not available for this counter/timer mode.
2061	Unable to convert Coord to Int32.
2062	Unable to convert Complex to Int32.
2063	Unable to convert PComplex to Int32.
2064	Unable to convert Spectrum to Int32.
2065	Unable to convert Enum to Int32.
2066	Unable to convert Text to Int32.
2067	Unable to convert Coord to Real.
2068	Unable to convert Complex to Real.
2069	Unable to convert PComplex to Real.
2070	Unable to convert Spectrum to Real.
2071	Unable to convert Enum to Real.
2072	Unable to convert Text to Real.
2073	Unable to convert Coord to Text.
2074	Unable to convert Complex to Text.
2075	Unable to convert PComplex to Text.
2076	Unable to convert Spectrum to Text.
2077	Requested Scan Mode not supported by this device.
2078	Device driver version <version number> is invalid. The device driver must be version 2.1 or greater.

2079	OLDAAPI.DLL version <version number> is invalid. OLDAAPI.DLL must be version 2.11 or greater. Reinstall DT VPI to install the correct version.
2080	OLMEM.DLL version <version number> is invalid. OLMEM.DLL must be version 2.12 or greater. Reinstall DT VPI to install the correct version.
2081	Driver underrun error. DT VPI could not process Windows messages. This is typically caused when a single object execution is too slow.

DT-Open Layers Errors

The following DT-Open Layers errors are generated from the device driver. For a description of VEE errors, refer to your VEE documentation.

Error Number	Description
0	Operation complete; no error.
1	Invalid subsystem capability requested.
2	Illegal element type specified.
3	Illegal subsystem specified.
4	Insufficient DMA channels are available to perform the requested operation.
5	The channel-gain list size is out of range.
6	Illegal channel-gain list entry specified.
7	The specified channel is out of range.
8	Illegal channel type requested.
9	Illegal encoding value requested.
10	Illegal trigger source requested.
11	Illegal resolution requested.
12	Illegal clock source value requested.
13	The specified clock frequency is out of range.
14	Illegal pulse type requested.
15	The pulse width is out of range.
16	Illegal counter mode requested.

Error Number	Description
17	Illegal cascade mode requested.
18	Illegal data flow requested.
19	Invalid window handle specified.
20	The requested subsystem is already in use.
21	The requested subsystem is not in use.
22	The subsystem is already running.
23	The subsystem has no channel list.
24	The subsystem has no gain list.
25	The subsystem has no filter list.
26	The subsystem has not been configured.
27	Single value operation was attempted while in continuous data flow mode (or vice versa).
28	Stop issued when subsystem wasn't running.
29	Requested range cannot be set.
30	Illegal subsystem capability specified.
31	Illegal device capability specified.
32	Illegal range index requested.
33	Illegal filter index requested.
34	Illegal gain index requested.
35	Illegal wrap mode requested.
36	Request not supported by subsystem.
37	The specified external clock divider is out of range.
38	Illegal gate type specified.

Error Number	Description
39	Illegal device handle specified.
40	Illegal subsystem handle specified.
41	The requested subsystem cannot be allocated.
42	The requested subsystem cannot be deallocated.
43	List could not be freed because buffers remained on the transfer list.
44	Another subsystem on the device is already running.
45	Channel list has been filled incorrectly.
46	Selected clock and trigger sources may not be used together.
47	Driver could not allocate needed memory.
48	Triggered scan is not available in single value mode.
49	External clock is not available in single value mode.
50	Illegal resolution index specified.
60	The A / D subsystem has a trigger error.
61	The A / D subsystem has an overrun error.
62	The D / A subsystem has a trigger error.
63	The D / A subsystem has an underrun error.
64	There are not any buffers on the ready queue.
65	Invalid CPU, DT-Open Layers requires an 80386 or above.
66	Invalid mode of Windows 3.1, DT-Open Layers requires enhanced mode.
67	The requested driver cannot be opened.
68	Cannot enumerate specified subsystem capability.

Error Number	Description
69	Dass procedure callback is NULL.
70	Invalid enumeration procedure.
71	A window handle has not been specified.
72	Specified subsystem cannot be cascaded.
73	Configuration settings read from SYSTEM.INI are invalid.
74	Driver could not allocate needed memory.
75	Driver could not allocate channel list.
76	Driver could not allocate gain list.
77	Driver could not allocate filter list.
78	No DT-Open Layers data acquisition boards are installed.
79	DMA cannot be used with multiple channels on this subsystem.
80	Invalid pulse type specified for this counter/timer mode.
81	Gain list set up incorrectly. See board's driver documentation.
82	Read events issued when subsystem not set for count mode.
83	Invalid string pointer.
84	Invalid polled I/O combination.
85	Invalid scan/trigger combination.
86	Invalid gain.
87	Invalid parameters for olDaMeasureFrequency<#010a>both the hWnd and the lpfnMeasureProc parameters are NULL.
88	Invalid count duration period specified.

Error Number	Description
89	Invalid queue specified.
90	Invalid retrigger rate specified.
91	No command response from hardware.
92	Hardware command sequence error.
93	Hardware data sequence error.
94	Cannot allocate timer data for driver.
95	Invalid timer handle.
96	Invalid timer mode.
97	Invalid timer frequency.
98	Invalid timer callback procedure.
99	Invalid timer DMA buffer size.
100	Illegal synchronous digital I/O value requested.
101	Cannot allocate simultaneous start list.
102	Illegal simultaneous start list handle specified.
103	Invalid subsystem handle on simultaneous start list.
104	No subsystem handles on simultaneous start list.
105	The subsystem has no channel inhibit list.
106	The subsystem has no digital I/O list.
107	The subsystem has not been prestarted.
108	Invalid notification procedure.
109	Invalid DT-Connect transfer count.
110	Invalid DT-Count transfer size.

Error Number	Description
111	Cannot allocate channel inhibit list.
112	Cannot allocate digital I/O list.
113	Channel inhibit list has been filled incorrectly.
114	DASS handle is already on simultaneous start list.
115	Cannot allocate range list.
116	No range list.
117	No buffers in process.
118	Additional required subsystem in use.
119	Wrap mode mismatch.
200	Cannot allocate a buffer control block for the requested data buffer.
201	Cannot allocate the requested data buffer.
202	Invalid buffer handle (HBUF) specified.
203	Could not lock specified HBUF.
204	Specified HBUF is locked.
205	Specified HBUF is already on a different list.
206	Reallocation of a buffer control block was unsuccessful.
207	Reallocation of the data buffer was unsuccessful.
208	Invalid sample size specified.
209	Cannot allocate buffer transfer list.
210	Invalid list handle specified.
211	Cannot free list because buffer's name has not yet been removed.

Error Number	Description
212	Specified HBUF is not locked.
213	Invalid DMA channel specified.
214	DMA channel is already in use by another subsystem.
215	Invalid interrupt level requested.
216	The requested interrupt level is already in use by another DT-Open Layers subsystem.
217	There are no samples in the specified buffer.
218	There are too many samples specified for the buffer.
219	Buffer is too small for requested copy operation.

File Formats

Single Channel Format 268
Multi-Channel Format 269
Channel-Gain List Entries 271

Single Channel Format

This section contains reference information about the file header used by the A/D Config and D/A Config objects for disk I/O. The format discussed is VEE binary format. As listed in the following table, the header is 124 bytes long and is followed by the data.

headersize Int16	2 bytes specifying the header size in bytes.
version Int16	2 bytes specifying the version of the file header.
boardname Char:40	40 bytes specifying the board name as defined by the user when the driver was installed. The board name is a null terminated ASCII string.
subsysname Char:40	40 bytes specifying the subsystem name. The subsystem name is a null terminated ASCII string.
datawidth Int16	2 bytes specifying the size of each data sample in bytes.
resolution Int16	2 bytes specifying the resolution of the board in bits, typically 12 or 16.
encoding Int16	2 bytes specifying the encoding of the data. A value of 0 is binary encoding and a value of 1 is twos complement encoding.
minvolts Real64	8 bytes specifying the minimum voltage of the voltage range. This value is the minimum voltage of the device's range divided by the gain.
maxvolts Real64	8 bytes specifying the maximum voltage of the voltage range. This value is the maximum voltage for the device's range divided by gain.
channel Int16	2 bytes specifying the channel number.
frequency Real64	8 bytes specifying the rate.
timedate Real64	8 bytes specifying the time and date the file was created. The value is the number of seconds since 00:00 hours 1 January 0001 AD given in UTC.

Multi-Channel Format

This section contains reference information about the file header used by the A/D Config object for multi-channel files. The header is a minimum of 128 bytes long and additionally includes the channel gain list entries and data.

headersize Int16	2 bytes specifying the header size in bytes. This number includes the total size of the channel gain list entries as well.
version Int16	2 bytes specifying the version of the file header. Changed for DT VPI 4.0.
boardname Char:40	40 bytes specifying the board name as defined by the user when the driver was installed. The board name is a null terminated ASCII string.
subsysname Char:40	40 bytes specifying the subsystem name. The subsystem name is a null terminated ASCII string.
datawidth Int16	2 bytes specifying the size of each data sample in bytes.
resolution Int16	2 bytes specifying the resolution of the board in bits, typically 12 or 16.
encoding Int16	2 bytes specifying the encoding of the data. A value of 0 is binary encoding and a value of 1 is twos complement encoding.
minvolts Real64	8 bytes specifying the minimum voltage of the voltage range.
maxvolts Real64	8 bytes specifying the maximum voltage of the voltage range.
numchannels Int16	2 bytes specifying the number of channels in the channel-gain list.

frequency	Real64	8 bytes specifying the aggregate clock rate.
timedate	Real64	8 bytes specifying the time and date the file was created. The value is the number of seconds since 00:00 hours 1 January 0001 AD given in UTC.
retriggerfreq	Real64	8 bytes specifying the retrigger rate if triggered scan mode is used. Otherwise, the value is zero.

Channel-Gain List Entries

These channel gain list entries are repeated “numchannels” times.

channel Int16	2 bytes specifying the channel in the channel-gain list.
gain Int16	2 bytes specifying the gain applied to the previous channel.

Product Support

General Checklist 274
Service and Support. 275
Glossary 279

General Checklist

Should you experience problems using DT VPI, please follow these steps:

1. Read all the appropriate sections of this manual. Make sure that you have added any “Read This First” information to your manual and that have used this information.

2. Check your distribution disk for a README file. If present, this disk will include the latest installation and configuration information.

3. Check that you have installed and configured your board properly. (See the instructions in Chapter 1, "Installing DT VPI," in this manual.)

4. Check that you have installed your hardware properly. (See the installation documentation for your board.)

5. Check that you have installed any other software properly. (See the installation documentation for the associated software.)

6. Search the DT Knowledgebase in the Support section of the Data Translation web site (at www.datatranslation.com) for an answer to your problem.

If you are still having problems, please follow the instructions provided in the next section, “Service and Support.”

Service and Support

If you have difficulty using DT VPI, Data Translation's Technical Support Department is available to provide prompt technical assistance. Support upgrades, technical information, and software are also available.

All customers can always obtain the support needed. The first 90 days are complimentary, as part of the product's original warranty, to help you get your system running. Customers who call outside of this time frame can either purchase a support contract or pay a nominal fee (charged on a per-incident basis).

For "priority support," purchase a support contract. Support contracts guarantee prompt response and are very affordable; contact your local sales office for details.

Refer to the Data Translation Support Policy located at the end of this manual for a list of services included and excluded in our standard support offering.

Telephone Technical Support

Telephone support is normally reserved for original warranty and support-contract customers. Support requests from non-contract or out-of-warranty customers are processed after requests from original warranty and support-contract customers.

For the most efficient service, please complete the form on page 277 and be at your computer when you call for technical support. This information helps to identify specific system and configuration-related problems and to replicate the problem in house, if necessary.

You can reach the Technical Support Department by calling (508) 481-3700 x1401.

If you are located outside the USA, call your local distributor. The name and telephone number of you nearest distributor are provided in your Data Translation catalog.

If you are leaving a message to request a support call, please include the following information:

- Your name (please include proper spelling),
- Your company or organization (please include proper spelling),
- A phone number,
- An e-mail address where you can be reached,
- The hardware/software product you need help on,
- A summary of the issue or question you have,
- Your contract number, if applicable, and
- Your product serial number or purchase date.

Omitting any of the above information may delay our ability to resolve your issue.

Information Required for Technical Support

Name:_____Phone_____
Contract Number: _____
Address: _____
_____

Data Translation hardware product(s): _____
serial number:_____
configuration: _____
Data Translation software - SPO number:_____ _____
serial number:_____ version:_____
PC make/model: _____
operating system: _____ version:_____
Windows version: _____
processor: _____ speed:_____
RAM: _____ hard disk space:_____
network/number of users: _____ disk cache:_____
graphics adapter: _____ data bus:_____

I have the following boards and applications installed in my system:_____
_____
_____
I am encountering the following problem(s):_____
_____
_____
_____
_____
and have received the following error messages/codes: _____
_____
_____
I have run the board diagnostics with the following results: _____
_____
You can reproduce the problem by performing these steps:
1. _____
_____
2. _____
_____
3. _____
_____

E-Mail and Fax Support

You can also get technical support by e-mailing or faxing the Technical Support Department:

- **E-mail**: You can reach Technical Support at the following address: tsupport@datx.com

  Ensure that you provide the following minimum information:

  – Your name,

  – Your company or organization,

  – A phone number,

  – An email address where you can be reached,

  – The hardware/software product you need help on,

  – A summary of the issue you are experiencing,

  – Your contract number, if applicable, and

  – Your product serial number or purchase date.

  Omitting any of the above information may delay our ability to resolve your issue.

- **Fax**: Please photocopy and complete the form on page 277, then fax Technical Support at the following number: (508) 481-8620.

Support requests from non-contract and out-of-warranty customers are processed with the same priority as telephone support requests.

World-Wide Web

For the latest tips, software fixes, and other product information, you can always access our World-Wide Web site free of charge at the following address: http://www.datatranslation.com

Glossary

This glossary defines several terms used to name or describe VEE and DT VPI features.

Activate
1. To send a container to a terminal. See also *Container* and *Terminal*.

2. The action that resets the context of a UserObject before it operates each time. See also *Context* and *PreRun*.

Aliasing
In data acquisition, an error condition arising when a signal is sampled at too slow a rate, resulting in a misrepresentation of the analog signal.

Array
A data shape that contains a systematic arrangement of data items in one or more dimensions. The data items are accessed via indexes. See also *Data Shape*.

Analog-to-Digital Conversion
The process of sampling an analog signal and converting it to digital values.

Asynchronous
In an asynchronous operation, a device operates without a common signal to synchronize events. Rather, the events occur at unspecified times. VEE control pins are asynchronous.

Auto Execute
An option on the object menus of the data constant objects. When Auto Execute is set, the object operates when its value is edited.

Base Frequency
See *Frequency Source*.

Bitmap
A bit pattern or picture. In VEE you can display a bitmap on an icon.

Buffer
An area in memory where information is stored temporarily.

Button
1. A button on a mouse.

2. A graphical object in VEE that simulates a real-life pushbutton and appears to pop out from your screen. When you "press" a button in VEE, by clicking on it with the mouse, an action occurs.

Cascading Menu
A sub-menu on a pull-down or pop-up menu that provides additional selections.

Channel List
A list of the channels used in acquiring analog data or outputting analog data. Each channel is connected to an external analog device, which either provides a signal to be acquired or accepts an analog signal. The channels are always used in order, from lowest to highest.

Checkbox
A recessed square box on VEE menus and dialog boxes that allows you to select a setting. To select a setting, click on the box and a checkmark appears in the box to indicate a selection has been made.To cancel the setting, simply click on the box again.

Click
To press and release a mouse button quickly. Clicking usually selects a menu feature or object in the VEE window. See also *Double-Click* and *Drag*.

Compiled Function
A user-defined function created by dynamically linking a program, written in a programming language such as C, into the VEE process. For Windows, the user must create a DLL (Dynamically Linked Library) file and a definition file. The Import Library object attaches the shared library or DLL to the VEE process and parses the definition file declarations. The Compiled Function can then be called with the Call Function object, or from certain expressions. See also *User Function*.

Component
A single instrument function or measurement value in a VEE instrument panel or component driver. For example, a voltmeter driver contains components that record the range, trigger source, and latest reading. See also *Component Driver*, *Driver Files*, *State*, and *Instrument Panel*.

Composite Data Type
A data type that has an associated shape. See also *Data Shape* and *Data Type*.

Container
The package that is transmitted over lines and is processed by objects. Each container contains data, the data type, and the data shape.

Context
A level of the work area that can contain other levels of work areas (such as nested UserObjects), but is independent of them.

Control Pin
An asynchronous input pin that transmits data to the object without waiting for the object's other input pins to contain data. For example, control pins in VEE are commonly used to clear or autoscale a display.

Cursor
A pointer in an entry field that shows where alphanumeric data will appear when you type information from the keyboard.

Data Field
The field within a transaction specification in which you specify either the expression to be written (WRITE transactions), or the variable to receive data that is read (READ transactions). See also *Transactions*.

Data Flow
The flow of data through and between VEE objects. Data flows from left to right through objects, but an object does not execute until it has data on all of its data input pins. Data is propagated from the data output pin of one object to the data input pin of the next object. Data flow is the chief factor that determines the execution of a VEE program.

Data Input Pin
A connection point on the left side of an object that permits data to flow into the object.

Data Output Pin
A connection point on the right side of an object that propagates data flow to the next object and passes the results of the first object's operation on to the next object.

DataSet
A collection of Record containers saved into a file for later retrieval. The To DataSet object collects Record data on its input and writes that data to a named file (the DataSet). The From DataSet object retrieves Record data from the named file (the DataSet) and outputs that data as Record containers on its Rec output pin. See also *Record*.

Data Shape
A pre-defined structure that defines how data is grouped together (for example, an array).

Data Type
A pre-defined structure that determines how data is organized and treated by VEE (for example, Real or Complex).

DDE Client
An application that requests a DDE conversation with another application. VEE has only client capabilities. See also *DDE (Dynamic Data Exchange)*.

DDE (Dynamic Data Exchange)
A communication mechanism that allows VEE to communicate with other Windows applications that support DDE. VEE can send data to, and receive data from, such applications. Also, VEE can execute commands in the other application. Examples of Windows applications that support DDE are Microsoft Excel and Microsoft Word for Windows.

DDE Server
An application that responds to a request for a DDE conversation from another application. VEE does not have DDE server capabilities. See also *DDE (Dynamic Data Exchange)*.

Default Button
The button in a dialog box that is activated by default if Enter or Return is pressed, or the selection is double-clicked. The default button has a recessed border.

Demote
To convert from a data type that contains more information to one that contains less information. See also *Data Type* and *Promote.*

Detail View
The view of a VEE program that shows all the objects and the lines between them.

Device
An instrument attached to or plugged into an HP-IB, RS-232, GPIO, or VXI interface. Specific VEE objects such as the Direct I/O object send and receive information to a device.

Dialog Box
A secondary window displayed when VEE requires information from you before it can continue. For example, a dialog box may contain a list of files from which you may choose.

Digital-to-Analog Conversion
The process of converting digital values to analog voltage values.

DLL (Dynamically Linked Library)
A collection of functions written in C that can be called from VEE. DLLs can be created by experienced C programmers using tools available from Microsoft and Borland.

DMA (Direct Memory Access)
A method of transferring data from a data acquisition board to memory that circumvents the CPU. DMA is much faster than the older polled I/O and interrupts methods. Some data acquisition boards use two DMA channels (use one channel while the other fills with data, then switch), while other boards can operate with a single DMA channel and still provide gap-free data acquisition.

Double-Click
To press and release a mouse button twice in rapid succession. Double-clicking is usually a short-cut to selecting and performing an action. For example, double-clicking on a file name from **File** ⇨ **Open** will select the file and open it.

Drag
To press *and continue to hold down* a mouse button while moving the mouse. Dragging moves something (for example, an object or scroll slider).

Entry Field
A field that is typically part of a dialog box or an editable object, and which is used for text entry. An entry field appears recessed. For example, the open view of the For Range object has entry fields where you type values that specify the beginning, ending, and step values.

Error Message
Information that appears in an error dialog box, explaining that a problem has occurred.

Error Pin
A pin that traps any errors that occur in an object. Instead of getting an error message, the error number is output on the error pin. When an error is generated, the data output pins are not activated.

Execute
The action of a program, or parts of a program, running.

Execution Flow
The order in which objects operate. See also *Data Flow*.

Expression
An equation in an entry field that can contain the input terminal names and any Math or AdvMath functions. An expression is evaluated at run-time. Expressions are allowed in Formula, If/Then/Else, Get Values, Get Field, Set Field, and Sequencer objects, and in I/O transaction objects.

Feature
An item on a menu that you select to cause a particular action to occur (for example, to open a file), or to get a particular object.

Feedback
A continuous thread path of sequence and/or data lines that uses values from the previous execution to change values in the current execution.

Flow
See *Data Flow* and *Execution Flow*.

Frequency Source
Either an on-board oscillator or a user-supplied signal that provides a clock used in data acquisition.

Function
The name and action of objects where the output is a function of the input. These objects are located under **Math** or **AdvMath** menus and may be used in the Formula object. For example, sqrt(x) is a function; + is not.

Global Variable
A named variable that is set globally, and which can be used by name in any context of a VEE program. For example, a global variable can be set with Set Global in the root context of the program, and can be accessed by name with Get Global or from certain expressions within the context of a UserObject. However, a local variable with the same name as the global variable takes precedence in an expression.

Grayed Feature
A menu feature that is displayed in gray rather than black, indicating that the feature is not active or not available.

Group Window
A group window in Microsoft Windows is a window that contains icons for a group of applications. Each icon starts an application in the group.

Hertz (Hz)
Samples per second. For example, if an analog signal is sampled 1000 times every second, its sampling rate is 1000 Hz, or 1 kHz.

Highlight
1. The colored band or shadow around an object that provides a visual cue to the status of the object.

2. The change of color on a menu feature that indicates you are pointing to that feature.

Host
To begin a thread or subthread. For example, the subthread that is hosted by For Count is the subthread that iterates.

Hypertext
A system of linking topics so that you can jump to a related topic when you want more information. In on-line help systems, typically hypertext links are designated with underlined text. When you click on such text, related information is presented.

Icon
The small, graphical representation of a VEE object, such as the representation of an instrument, a control, or a display.

Interrupt
A signal that requires immediate attention that may suspend a process, such as the execution of a computer program. An interrupt is usually caused by an event external to that process. After the interrupt is serviced, the process may be resumed.

Label
The text area or name on an icon or button that identifies that object or button.

Library
A collection of often-used objects or small programs grouped together for easy access.

Line
A link between two objects in VEE that transmits data containers to be processed. See also *Subthread* and *Thread*.

Main Menu
The menus located in the VEE menu bar. The main menus may be opened by clicking or dragging on the menu titles in the menu bar.

Mapping
To associate a set of independent values with an array, when the array is a function of the values.

Maximize
To enlarge a window to its maximum size. In VEE, the UserObject has a maximize button.

Menu
A collection of features that are presented in a list. See also *Cascading Menu*, *Main Menu*, *Object Menu*, *Pop-Up Menu*, and *Pull-Down Menu*.

Menu Bar
A rectangular bar at the top of the VEE window that contains titles of the pull-down, main menus from which you select features.

Menu Title
The name of a menu within the VEE menu bar. For example, **File** or **Edit**.

Minimize
1. To reduce an open view of an object to its smallest size—an icon.

2. To reduce a window to its smallest size—an icon.

Mouse
A pointing device that you move across a surface to move a pointer within the VEE window.

Mouse Button
One of the buttons on a mouse that you can click, double-click, or drag to perform a particular action with the corresponding pointer in the VEE window.

Network
A group of computers and peripherals linked together to allow the sharing of data and work loads.

Object
A graphical representation of an element in a program, such as an instrument, control, display, or mathematical operator. An object is placed in the work area and connected to other objects to create a program. Objects can be displayed as icons or as open views.

Object Menu
The menu associated with an object that contains features that operate on the object (for example, moving, sizing, copying, and deleting the object).

Open
To start an action or begin working with a text, data, or graphics file. When you select **Open** from VEE, a program is loaded into the work area.

Open View
The representation of a VEE object that is more detailed than an icon. Within the open view, you can modify the operation of the object and change the object's title.

Operate
The action of an object processing data and outputting a result. An object operates when its data and sequence input pins have been activated. See *Activate*.

Outline Box
A box that represents the outer edges of an object or set of objects and indicates where the object(s) will be placed in the work area.

Pacer Clock
Circuitry on a data acquisition board which sets the pace for analog-to-digital or digital-to-analog conversions. It consists of a frequency source of 400kHz to 10MHz and a divider network that steps the frequency source down to more usable values.

Palette
A set of colors and fonts that is supplied with VEE and used in your VEE environment.

Panel
Information displayed in the center of the object's open view. In a UserObject, the panel contains a work area. In a For Count object, the panel contains an entry field. Compare with *Panel View*.

Panel View
The view of a program in VEE that shows only those objects needed for the user to run the program and view the resultant data. You can create a panel view to meet the needs of your users.

Pin
An external connection point on an object to which you can attach a line.

Point
See *Sample*.

Pointer
The graphical image that maps to the movement of the mouse. A pointer allows you to make selections and provides you feedback on a particular process underway. VEE has pointers of different shapes that correspond to process modes, such as an arrow, crosshairs, and hourglass.

Pop-Up Menu
A menu that is raised by clicking the right mouse button. For example, you can raise the **Edit** menu by clicking the right mouse button in an empty area in the work area. Or you can raise the object menu by clicking the right mouse button on an inactive area of an object.

PostRun
The set of actions that are performed when the program is stopped.

PreRun
The set of actions that resets the program and checks for errors before the program starts to run.

Priority Thread
A priority thread executes to completion blocking all other parallel threads from executing. Certain of the I/O objects for devices and interfaces will host a priority thread.

Program
In VEE, a graphical program that consists of a set of objects connected with lines. The program typically represents a solution to an engineering problem.

Promote
To convert from a data type that contains less information to one that contains more information. See also *Data Type* and *Demote*.

Propagation
The rules that objects and programs follow when they operate or run. See also *Data Flow*.

Pull-Down Menu
A menu that is pulled down from the menu bar when you position the pointer over a menu title and click the left mouse button.

Radio Button
A diamond-shaped button in VEE dialog boxes that allows you to select a setting that is mutually exclusive with other radio buttons in that dialog box. To select a setting, click on the radio button. To remove the setting, click on another radio button in the same dialog box.

Record
A data type that has named data fields which can contain multiple values. Each field can contain another Record container, a Scalar, or an Array. The Record data type has the highest precedence of all VEE data types. However, data cannot be converted to and from the Record data type through the automatic promotion / demotion process. Records must be built / unbuilt using the using Build Record and UnBuild Record objects.

Restore
To return a minimized window or an icon to its full size as a window or open view by double-clicking on it.

Run
To start the objects on a program or thread operating.

Sample
1. A representation of an analog signal obtained by looking at single values in the signal. The values are close enough together in time to accurately recreate the signal.

2. A single value in such a representation.

Sampling Frequency
See “Sampling Rate”.

Sampling Rate
The frequency at which an analog signal is sampled, usually in Hertz (Hz). For example, if an analog signal is sampled 1000 times every second, the sampling rate is 1000Hz, or 1 kHz.

Save
To write a file to a storage device, such as a hard disk, for safekeeping.

Scalar
A data shape that contains a single value. See also *Data Shape.*

Schema
The structure or framework used to define a data record. This includes each field's name, type, shape (and dimension sizes), and mapping.

Screen Dump
A graphical printout of a window or part of a window.

Scroll
The act of using a scroll bar either to move through a list of data files or other choices in a dialog box, or to pan the work area.

Scroll Arrow
An arrow that, when clicked on, moves you through a list of data files or other choices in a dialog box, or moves the work area.

Scroll Slider
A rectangular bar that, when dragged, moves you through a list of data files or other choices in a dialog box, or moves the work area.

Select
To choose an object, an action to be performed, or a menu item. Usually you select by clicking with your mouse.

Select Code
A number used to identify the logical address of a hardware interface. For example, the factory default select code for most HP-IB interfaces is 7.

Selection
1. A menu selection (feature).

2. An object or action you have selected in the VEE window.

Sequence Input Pin
The top pin of an object. When connected, execution of the object is held off until the pin receives a container.

Sequence Output Pin
The bottom pin of an object. When connected, this output pin is activated when the object and all data propagation from that object finishes executing.

Sequencer
An object that controls execution flow through a series of sequence transactions, each of which may call a "User Function" or "Compiled Function." The sequencer is normally used to perform a series of tests by specifying a series of sequence transactions.

Sleep
An object sleeps during execution when it is waiting for an operation or time interval to complete, or for an event to occur. A sleeping object will allow other parallel threads to run concurrently. Once the event, time interval, or operation occurs, the object will execute, allowing execution to continue.

Step
The action of operating one object at a time. An arrow points to the object that will operate next.

Subsystem
Functional sections on a data acquisition hardware. Each subsystem performs a particular function, such as analog input, analog output, digital input, digital output, or counter/timer functions. Prior to use, a subsystem must be configured by a subsystem configuration object in VEE.

Subsystem Configuration Object
One of the five data acquisition objects that are used to configure a subsystem on a data acquisition board. The subsystem configuration objects are:

- A/D Config
- D/A Config
- Digital In Config
- Digital Out Config
- C/T Config

Subsystem Handle
A number identifying the subsystem of interest to data acquisition objects. After a subsystem configuration object is run, a handle for that subsystem exists. This handle is passed as input to any data acquisition object that needs to interact with that subsystem.

Terminal
The internal representation of a pin that displays information about the pin and the data container held by the pin. Double-click the terminal to view the container information.

Thread
A set of objects connected by solid lines in a VEE program. A program with multiple threads can run all threads simultaneously.

Title Bar
The rectangular bar at the top of the open view of an object or window, which shows the title of the object or window.

Tool Bar
The rectangular bar at the top of the VEE window which provides buttons to control VEE programs, open and save files, and many other handy tools.

Transaction
The specifications for input and output (I/O) used by certain objects in VEE. These include the To File, From File, and Sequencer objects. Transactions appear as English-like phrases listed in the open view of these objects.

User-Defined Function
VEE allows two types of user-defined functions: the *User Function* and *Compiled Function*.

Triggered Scan
A kind of data acquisition where data from several channels are sampled each time a trigger pulse is received. The data for each channel is in the same time domain, over which you have control. See Chapter 4 for more information.

UserFunction
A user-defined function created from a UserObject by selecting **Make UserFunction** from a UserObject's user menu. The UserFunction exists in background, but provides the same functionality as the original UserObject.

You can call a UserFunction with the Call Function object, or from certain expressions. A UserFunction can be created and called locally, or it can be saved in a library and imported into a VEE program with Import Library. See also *Compiled Function* and *UserObject*.

User Interface
The part of an application that permits a user and the application to communicate with each other to perform certain tasks. VEE uses a graphical user interface, which includes windows, menus, dialog boxes, and objects.

UserObject
An object that can encapsulate a group of objects to perform a particular purpose within a program. A UserObject allows you to use top-down design techniques when building a program, and to build user-defined objects that can be saved in a library and reused.

View
See *Detail View*, *Icon*, *Open View*, and *Panel View*.

Wait
See *Sleep*.

Window
A rectangular area on the screen that contains a particular application program, such as VEE.

Work Area
The area within the VEE window or the open view of a UserObject where you group objects together. When you open a program, it is loaded into the work area.

XEQ Pin
A pin that forces the operation of the object, even if the data or sequence input pins have not been activated. See also *Control Pin*, *Data Input Pin*, and *Sequence Input Pin*.

Index

Numerics

727.VEE example program 246

A

A / D Config object 26, 44, 51, 81
about-trigger acquisition 110
accessing unsecured UserObjects 240
Acq Status object 100, 122, 142, 173, 183
acquiring data 43
  a single value from a single channel 101
  continuous analog data 48, 96
  from a single channel 92
  from multiple channels 94
  in burst mode 104
  post-trigger 111
  pre-trigger 109, 111
  single analog value 50
  single digital input value 189
ADBURST.VEE example program 104, 246
ADCOLUMN.VEE example program 246
adding an amplitude input 40
adding configuration objects 72
adding noise to a waveform 38
adding objects 14, 31, 44, 51, 61, 67
adding terminals 23
adding user input 74
ADSIMPLE.VEE example program 25, 246
ADSINGLE.VEE example program 103, 246
advanced functions
  CompensationVolts 242
  dtv.ConfigSubSys 229
  dtv.GetChannelType 230
  dtv.GetCriticalThreshold 231
  dtv.GetData 232
  dtv.GetEncoding 235
  dtv.GetFileSize 138, 235
  dtv.GetMixedBufferSize 236
  dtv.GetPreTrigData 233
  dtv.GetResolution 187, 237
  dtv.GetSingleValueEx 238
  dtv.GetTriggerThreshold 239
  dtv.PutData 234
  dtv.PutSingleValueEx 238
  dtv.SetChannelType 230
  dtv.SetCriticalThreshold 163, 231
  dtv.SetMixedBufferSize 236
  dtv.SetResolution 190, 192, 237
  dtv.SetTriggerThreshold 239
  ThAverage 242
  VoltsToDegrees 243
aggregate sampling rate 117
aliasing 84
ALLCHAN.VEE example program 138, 246
Alloc Real object 75
Allocate Array object 173
Allow Gaps data mode 99, 142, 161, 183

analog input
  acquiring a burst of data 104
  acquiring a single value 101
  acquiring data continuously 96
  acquiring data from a single channel 92
  acquiring data from multiple channels 94
  building a channel list 83
  configuring 81
  displaying data 93
  external trigger 107
  filtering a signal 85
  optimizing 142
  pre-trigger data 109
  running with analog output 194
  selecting a board 82
  selecting a clock 88
  selecting a gain 86
  selecting a sampling rate 84
  selecting a trigger 88
  selecting an input range 87
  selecting an input type 87
  sending through DDE 139
  specifying a subsystem handle 89
  starting an operation 91
  triggered scan 112
  using C/T as a trigger 212
  using C/T signal as pacer clock 209
  writing to disk files 125, 129
analog output
  burst mode 166
  configuring 147
  configuring the channel list 149
  connecting external devices 147
  continuous output 159
  critical threshold 162, 163
  optimizing 183
  outputting a single value 164
  overview 146
  reading from disk 177
  running with analog input 194
  selecting a board 148
  selecting an output range 153
  selecting an output rate 150
  selecting an output type 153
  specifying a subsystem handle 154
  starting an operation 155
analog-to-digital conversion xii
ASCII file format 133
ATLABFIL.VEE example program 246

B

breakpoints 226
Build Waveform object 56
building a waveform 56
burst mode 104, 108, 142, 162
  configuring 105, 166
  optimizing analog output 183
  using when writing data to file 131
bus master 195
buttons
  Configure 45, 52, 62, 68, 71
  Cut 55, 71
  Delete Line 38, 73, 74
  Line Probe 26
  minimize 18
  New 43, 60
  Open 70
  Print 36
  Run 33
  Save 36, 54, 58, 65, 69, 77
  Show Data Flow 25

Show Execution Flow 26
Step 226
Stop 48, 54, 69, 97

C

C/T Config object 209
CAL2801.VEE example program 246
CAL2812.VEE example program 246
CAL2821.VEE example program 246
CALCJC.VEE example program 246
Calculate Actual Rates 142, 183
Call Function object 229
changing object parameters 34
changing the configuration 70
changing the title of an object 20
channel list 83, 92, 149
  disabling automatic build of 138
channel type 230
channel types 87
channel-gain list entries 271
channels
  states of 100, 162
  writing to a file 133
clock
  configuring 88
  configuring C/T as 210
  external frequency 170, 204, 208
  source 170, 204, 208
  using counter/timer as 209
Close Subsystem object 122, 142, 183
CNTEVENT.VEE example program 246
COHERENCE.VEE example program 246
collecting data 55
Collector object 55
CompensationVolts 242
Configure button 45, 52, 62, 68, 71
configuring
  analog input 81
  analog output 147
  board selection 82
  burst mode 105, 166
  changing the subsystem 117, 170
  clock source 88
  counter/timer as a trigger 213
  counter/timer as clock 210
  counter/timer to count events 216
  counter/timer to measure frequency 219
  digital I/O 187
  external trigger source 107
  for slow sampling rates 138
  for VEE RunTime 224
  hardware filter frequency 85
  input channel list 83
  input gain value 86
  input range and type 87
  input sampling rate 84
  optimizing 142, 183
  output channel list 149
  output range and type 153
  output rate 150
  pre-trigger acquisition 110
  sampling rate 119
  to read from a file 178
  to write to a file 133
  to write to a single file 135
  trigger source 88
  triggered scan 112
configuring the A/D subsystem 45, 52
configuring the D/A subsystem 62
configuring the digital input port 68

connecting
  a function generator 4
  a screw terminal panel 3
  an oscilloscope 4
  digital inputs and outputs 187
  objects 23, 32
CONTFFT.VEE example program 247
continuous acquisition 96
continuous output 159
control bar 18
conventions used xviii
copying an object 16
count events 205
Counter object 199
counter/timers 204
  configuring as a trigger 213
  configuring to count events 216
  configuring to measure frequency 219
  modes 205
  optimizing 221
  starting with another subsystem 200
  using as a clock 209
  using as a trigger 212
  using to count events 216
  using to measure frequency 218
counting events 216
creating a simple program 31
critical threshold 162, 163, 175, 231
CTRIGGER.VEE example program 212, 247
Cut button 55, 71
cutting an object 17

D

D/A Config object 61, 147
DABURST.VEE example program 166, 247
DACONT.VEE example program 153, 159, 247
DAFILE.VEE example program 177, 247
DASIMPLE.VEE example program 156, 247
DASINGLE.VEE example program 164, 247
data flow 22, 25
  changing with sequence pins 27
  rules 27
data input pins 22
data mode 99, 142, 161, 162, 163, 183, 231
  Allow Gaps 131
data output pins 22
data queues 99, 161
  reading a file as 177
DC-outputting 164
DDE
  receiving data from 180
  sending analog data to 139
DDE.VEE example program 141, 182, 247
delay 207
Delay object 199
Delete Line button 38, 73, 74
deleting an object 17
deleting terminals 24
device drivers 225
differential channels 87, 154
digital I/O 66, 191
  configuring 187
  connecting external devices 187
Digital In Config object 67, 187

Digital Out Config object 187
digital-to-analog conversion 146
DIN.VEE example program 189, 192, 247
DIO.VEE example program 190, 247
DMA transfer 195, 201
documenting your program 226
Done state 100, 162
DOUT.VEE example program 191, 247
dragging an object 15
DT VPI on-line help 11
dtv.ConfigSubSys 229
dtv.GetChannelType 230
dtv.GetCriticalThreshold 231
dtv.GetData 232
dtv.GetEncoding 235
dtv.GetFileSize 235
dtv.GetFileSize function 138
dtv.GetMixedBufferSize 236
dtv.GetPreTrigData 233
dtv.GetResolution 237
dtv.GetResolution function 187
dtv.GetSingleValueEx 238
dtv.GetTriggerThreshold 239
dtv.PutData 234
dtv.PutSingleValueEx 238
dtv.SetChannelType 230
dtv.SetCriticalThreshold 231
dtv.SetCriticalThreshold function 163
dtv.SetMixedBufferSize 236
dtv.SetResolution 190, 192, 237
dtv.SetTriggerThreshold 239
DTVEE.DLL 229
DTVEE.VH 229
DTVFILE.VEE example program 131, 142, 247
duplicating an object 16
duration 208
duty cycle 205, 207
Dynamic Data Exchange 139, 180

E

EASYPK.VEE example program 247
e-mail support 278
Enum object 107
equipment 43, 50, 61, 66, 70
error handling 121
errors
  aliasing 84
  DT VPI 252
  DT-Open Layers 260
  handling 121, 175
  monitoring acquisition 124
  trapping 122, 176
  when reading from a file 175
  when writing to a file 122
event counting 216
example programs 245
  727.VEE 246
  ADBURST.VEE 104, 246
  ADCOLUMN.VEE 246
  ADSIMPLE.VEE 25, 92, 246
  ADSINGLE.VEE 103, 246
  ALLCHAN.VEE 138, 246
  ATLABFIL.VEE 246
  CAL2801.VEE 246
  CAL2812.VEE 246
  CAL2821.VEE 246
  CALCJC.VEE 246
  CNTEVENT.VEE 246
  COHERENCE.VEE 246
  CONTFFT.VEE 247
  CTRIGGER.VEE 212, 247

DABURST.VEE 166, 247
DACONT.VEE 153, 159, 247
DAFILE.VEE 177, 247
DASIMPLE.VEE 156, 247
DASINGLE.VEE 164, 247
DDE.VEE 141, 182, 247
DIN.VEE 189, 192, 247
DIO.VEE 190, 247
DOUT.VEE 191, 247
DTVFILE.VEE 131, 142, 247
EASYPK.VEE 247
FILENAME.VEE 247
HCHAN1.VEE 247
HCHAN2.VEE 247
MAGBIN.VEE 247
MEASURE.VEE 248
MONITOR.VEE 241, 248
MULTFILE.VEE 248
MULTI.VEE 95, 248
MULTIFILE.VEE 138
opening 25
PCLABFIL.VEE 248
PRETRIG.VEE 248
PSD.VEE 248
SERIALIO.VEE 248
SIMULTAN.VEE 198, 248
SLOWAD.VEE 248
TEMP727.VEE 241, 248
THRESH.VEE 248
TOFILE.EXE 248
TOFILE.VEE 124
TRANSAVG.VEE 248
TRIGGER.VEE 107, 249
TRIGSCAN.VEE 115, 249
Execute Program (PC) object 139
external trigger 100, 107
configuring C/T as 213
using C/T as a source 212

F

fax support 278
FFT(x) object 99
file formats
multi-channel 269
single channel 268
FILENAME.VEE example program 247
files
formats of 131, 133, 178, 268, 269
reading as a data queue 177
reading directly from 177
writing all channels to a single file 133
writing directly to 130
writing each channel to a file 133
writing multiple channels to a single file 135
writing to 125, 129
filters 85
flow, data 22, 25, 27
For Count object 55, 103, 127, 135
Formula object 200
frequency 207
external clock 170
frequency measurement 218
From File object 129, 177
function generator 4
Function Generator object 14, 31, 61, 151, 157

G

gain 86
Gap Free data mode 99, 161
  Allow Gaps data mode 163
gates 204, 206
gathering equipment 43, 50, 61, 66, 70
generate rate mode 205, 210
Get C/T Frequency object 118, 171
Get C/T Measure Duration object 118, 171
Get C/T Mode object 118, 171
Get C/T Pulse Width object 118, 171
Get Channel Buffer Size object 117, 170
Get Channel Filter object 118, 171
Get Clocking object 117, 170
Get Config objects 117, 170
Get Data Function object 142, 199
Get Data Mode object 118, 171
Get Data Panel object 26, 44, 94, 143
  matching to channel list 92
Get Multi-Channel Mode object 118, 138, 171
Get Range object 117, 170
Get Sample Clock Frequency object 171
Get Sampling Clock Frequency object 118
Get Sampling object 72, 117, 119, 170, 173, 199
Get Single Value object 51, 53, 55, 67, 101, 189
Get Trigger object 117, 170, 214, 225
getting a single value 50
glossary 279
graphs 93

H

handling errors 175
HCHAN1.VEE example program 247
HCHAN2.VEE example program 247
help
  starting DT VPI on-line help 11
  starting VEE on-line help 11

I

icon view 18
Import Library object 229
installing
  data acquisition boards 3
  device drivers 4
  DT VPI 5
  summary 2
  VEE 4
Integer Constant object 165
Integer object 51, 67

L

lessons
  acquiring a single analog value 50
  acquring analog data 43
  changing the configuration programmatically 70
  creating a simple program 31
  digital input and output 66
  modifying a simple program 38
  outputting analog data 60
Line Probe button 26
loading device drivers 225
Logging AlphaNumeric object 67, 93, 128

M

MAGBIN.VEE example program 247
measure frequency 205
MEASURE.VEE example program 248
measuring frequency 218
measuring temperatures 241
memory use 108, 169
menu bar 10
Mixed Data Buffer 97, 99, 108, 127, 161
modifying a program 38, 71
MONITOR.VEE example program 241, 248
moving objects 15
MULTFILE.VEE example program 248
MULTI.VEE example program 248
multi-channel file configuration 135
MULTIFILE.VEE example program 138

N

New button 43, 60
Noise Generator object 38, 40
Nyquist theorem 47, 84

O

object menu
  pictured 16
  selecting 15
objects
  A/D Config 26, 44, 51, 81
  Acq Status 100, 122, 142, 173
  adding 14, 31, 44, 51, 61, 67
  adding an error output pin to 122
  adding terminals 23
  Alloc Real 75
  Allocate Array 173
  Build Waveform 56
  C/T Config 209
  Call Function 229
  changing parameters 34
  changing the title 20
  changing object views 18
  Close Subsystem 122, 142, 183
  Collector 55
  connecting 23, 32
  control bar 18
  copying 16
  Counter 199
  cutting 17
  D/A Config 61, 147
  Delay 199
  deleting 17
  deleting terminals 24
  Digital In Config 67, 187
  Digital Out Config 187
  dragging 15
  duplicating 16
  Enum 107
  Execute Program (PC) 139
  FFT(x) 99
  For Count 55, 103, 127, 135
  Formula 200
  From File 129, 177
  Function Generator 14, 31, 61, 151, 157
  Get C/T Frequency 118, 171
  Get C/T Measure Duration 118, 171
  Get C/T Mode 118, 171
  Get C/T Pulse Width 118, 171
  Get Channel Buffer Size 117, 170
  Get Channel Filter 118, 171
  Get Clocking 117, 170

Get Data Function 142, 199
Get Data Mode 118, 171
Get Data Panel 26, 44, 92, 94, 143
Get Multi-Channel Mode 118, 138, 171
Get Range 117, 170
Get Sample Clock Frequency 171
Get Sampling 72, 117, 119, 170, 173, 199
Get Sampling Clock Frequency 118
Get Single Value 51, 53, 55, 67, 101, 189
Get Trigger 117, 170, 214, 225
iconizing 18
Import Library 229
in Get Config menu 117, 170
in Set Config menu 117, 170
Integer 51, 67
Integer Constant 165
Logging AlphaNumeric 67, 93, 128
moving 15
Noise Generator 38, 40
pasting 18
Put Data Function 155, 183
Put Data Panel 61, 155, 156, 174
Put Single Value 164, 191
Real Knob 41
Real Slider 74
renaming terminals 24
resizing 19
selecting object menus 15
Set C/T Frequency 118, 171
Set C/T Measure Duration 118, 171
Set C/T Mode 118, 171
Set C/T Pulse Width 118, 171
Set Channel Buffer Size 117, 170
Set Channel Filter 118, 171
Set Clocking 117, 170
Set Data Mode 118, 171
Set Multi-Channel Mode 118, 171
Set Range 117, 170
Set Sample Clock Frequency 171
Set Sampling 72, 117, 119, 170, 173, 199
Set Sampling Clock Frequency 118
Set Trigger 107, 117, 170, 214
Spectrum (Freq) 93
Start 27
Start Acq 91, 155, 174
Stop Acq 119, 122, 131, 162, 173
Strip Chart 51, 53, 93
To File 125
To/From DDE 140, 180
Until Break 48, 51, 67, 96
Waveform (Time) 26, 31, 44, 64, 93
XY Trace 128
one-shot mode 205
on-line help
DT VPI 11
VEE 11
Open button 70
open view 18
optimizing
analog input 142
analog output 183
counter/timers 221
oscilloscope 4
output rate 150
outputting
analog data 60
outputting data
a single analog value 164
a single digital value 191
analog burst 166

analog data from a single channel 156
continuous analog data 159
overview
data acquisition xi
data acquisition boards xiv
DT VPI xi

P

pacer clock xii, 50
panel view 224
pasting an object 18
PCI bus mastering 195
PCLABFIL.VEE example program 248
ping 56
pins 22
data input 22
data output 22
sequence input 22
sequence output pin 22
XEQ 56
Polled I/O 195
post-trigger acquisition 111
post-trigger source 110
PRETRIG.VEE example program 248
pre-trigger data 109
Print button 36
printing the screen 36
product support 274
propagation 25
PSD.VEE example program 248
pulse type 205
pulse width 207
Put Data Function object 155, 183
Put Data Panel object 155, 174
position in program 156
Put Data Panel objects 61
Put Single Value object 164, 191

Q

quitting VEE 13

R

Ready state 100, 161, 162
Real Knob object 41
Real Slider object 74
related documents xix
renaming terminals 24
resizing an object 19
Run button 33
Running state 100, 162
running two subsystems at once 194
running your program 33, 41, 47, 53, 58, 64, 69, 76

S

sampling rate xii, 84, 183
configuring for slow 138
requirements for writing to file 125, 130
setting for multi-channel file output 136
Save button 36, 54, 58, 65, 69, 77
saving your program 36, 49, 54, 58, 65, 69, 77
screen, printing 36
screw terminal panel 3, 218
securing a program 226
selecting a board 148
analog input 82
sequence input pin 22

sequence output pin 22
SERIALIO.VEE example program 248
service 275
Set C/T Frequency object 118, 171
Set C/T Measure Duration object 118, 171
Set C/T Mode object 118, 171
Set C/T Pulse Width object 118, 171
Set Channel Buffer Size object 117, 170
Set Channel Filter object 118, 171
Set Clocking object 117, 170
Set Config objects 117, 170
Set Data Mode object 118, 171
Set Multi-Channel Mode object 118, 171
Set Range object 117, 170
Set Sample Clock Frequency object 171
Set Sampling Clock Frequency object 118
Set Sampling object 72, 117, 119, 170, 173, 199
Set Trigger object 107, 117, 170, 214
sharing DMA resources 201
Show Data Flow button 25
Show Execution Flow button 26
Show Terminals option 22
Simple Driver 26
SIMULTAN.VEE example program 198, 248
single value 101, 164
Single Value transfer 195
single-ended channels 87, 154
sizing an object 19
SLOWAD.VEE example program 248
Spectrum (Freq) object 93
Start Acq object 91, 155, 174
Start object 27
starting
  analog input operations 91
  analog output operations 155
  two subsystems simultaneously 200
  VEE 8
states of channels 100, 162
Step button 226
Stop Acq object 119, 122, 131, 162, 173
  using with Get Config object 172
Stop button 48, 54, 69, 97
Strip Chart object 51, 53, 93
subsystem
  A/D 45
  changing configuration 117, 170
  closing 142
  configuring A/D 81
  configuring for VEE RunTime 224
  counter/timer 200
  running two simultaneously 194
  starting two simultaneously 200
subsystem handle 89, 154
subsystems xiii
support 275
  e-mail 278
  fax 278
  telephone 275
  World Wide Web 278
system requirements xv

T

technical support 275
  e-mail 278
  fax 278
  telephone 275
  World-Wide Web 278
telephone support 275

TEMP727.VEE example program 241, 248
temperature measurements 241
terminal area 40
terminal labels 22
terminals 22
  adding 23
  deleting 24
  renaming 24
ThAverage 242
thermocouple support 241
thermocouple UserFunctions 228
thread 99, 161
THRESH.VEE example program 248
title bar 10
titles, changing 20
To File object 125
To/From DDE object 140, 180
TOFILE.VEE example program 124, 248
tool bar 10
TRANSAVG.VEE example program 248
trapping errors 122, 176
trigger
  configuring 88
  external 107
  source 170
  threshold 239
TRIGGER.VEE example program 249
TRIGGER.VEE object 107
triggered scan 112
triggers xii
TRIGSCAN.VEE example program 115, 249
TTL signal 107

U

Until Break object 48, 51, 67, 96

V

VEE
  installing 4
  quitting 13
  starting 8
  starting DT VPI on-line help 11
  starting VEE on-line help 11
VEE (binary) file format 133, 178
  multi-channel 269
  single channel 268
VEE RunTime 224
VEE.IO file 225
veedoc utility 226
Volts vs.Time graph 93
VoltsToDegrees 243

W

Waiting state 162
Waveform (Time) object 26, 31, 44, 64, 93
waveform, building 56
work area 11
  clearing 14
working with objects 14
World-Wide Web 278
writing to files 125, 129

X

XEQ pin 56
XY Trace object 128

Data Translation Support Policy

Data Tr**an**slation, Inc. (Data Translation) offers support upon the following terms and conditions at prices published by Data Translation from time to time. Current price information is available from Data Translation, or its authorized distributor. If Licensee elects to obtain support services from Data Translation, Licensee must complete the Support Order Form attached hereto and submit to Data Translation the completed form, along with Licensee's purchase order for support. Support will only be provided for all (not less than all) Licensed Processors (as defined in the Data Translation Software License Agreement).

1. DEFINITIONS. Capitalized terms used herein and not otherwise defined shall have the meanings assigned thereto in the applicable Data Translation Software License Agreement (the Agreement). The following terms have the meanings set forth below:

Enhanced Release means a new release of any Product that contains new features and may contain corrections to previously identified errors. Enhanced Releases are designated in the tenths digit of the release designation (e.g., 1.2 is an Enhanced Release from 1.1.x).

Maintenance Release means a new release of any Product that contains corrections to previously identified errors. Maintenance Releases are designated in the hundredths digit of the release designation (e.g., 1.2.2 is a Maintenance Release from 1.2.1).

Major Release means a new version of any Product that involves major feature changes. Major Releases are designated in the ones digit of the release designation (e.g., 2.0, 3.0, etc., are Major Releases).

2. DATA TRANSLATION'S OBLIGATIONS. Subject to the terms of the Agreement, and this Support Policy, Data Translation will provide the following support services (Support Services) for the Products comprising the Software, as they may be used with the Licensed Processors:

(a) problem reporting, tracing and monitoring by interned electronic mail; (b) telephone support for problem determination, verification and resolution (or instruction as to work-around, as applicable) on a call-back basis during Data Translation's normal weekday business hours of 8:30 a.m. to 5 p.m. Eastern Time, excluding holidays; (c) one (1) copy of each Maintenance Release for the Products comprising the Software; (d) commercially reasonable efforts to diagnose and resolve defects and errors in the Software and Documentation; and (e) furnishing of the maintenance and technical support described above, for the current release and the immediately previous Enhanced Release of the Software. Support Services will be delivered in English. Enhanced Releases and Major Releases can be purchased by Licensee at a discount of twenty five percent (25%) off the then-current list prices for such releases.

3. EXCLUSIONS. Support Services do not include: (a) the provision of or support for Products other than those identified in the Agreement as to which the applicable license and support fees shall have been paid, including without limitation, compilers, debuggers, linkers or other third party software or hardware tools or components used in conjunction with any Product; (b) services required as a result of neglect, misuse, accident, relocation or improper operation of any Product or component thereof, or the failure to maintain proper operating and environmental conditions; (c) support for processors other than Licensed Processors or for Products modified by or on behalf of Licensee; (d) repair or restoration of any Software arising from or caused by any casualty, act of God, riot, war, failure or interruption of any electrical power, air conditioning, telephone or communication line or any other like cause.

Data Translation Support Policy

It is Licensee's responsibility to have adequate knowledge and proficiency with the use of the compilers and various software languages and operating systems used with the Products, and this Support Policy does not cover training of, or detailed direction on the correct use of these compilers, operating systems, or components thereof. On-site assistance shall not be provided hereunder, but may be available on a per call basis at Data Translation's then current rates (Specialized Application Support Charges) for labor, travel time, transportation, subsistence and materials during normal business hours, excluding holidays observed by Data Translation. The troubleshooting of faulty Licensee programming logic may also be subject to Specialized Application Support Charges and is not covered under this Support Policy. Direct authoring or development of customized application code is not provided hereunder but may be available on a per call basis upon payment of Specialized Application Support Charges.

4. LICENSEE'S OBLIGATIONS. Licensee agrees: (a) that the Designated Contact persons identified on the Support Order Form (or such other replacement individuals as Licensee may designate in writing to Data Translation) shall be the sole contacts for the coordination and receipt of the Support Services set forth in Section 2 of this Support Policy; (b) to maintain for the term of the support, an internet address for electronic mail communications with Data Translation; (c) to provide reasonable supporting data (including written descriptions of problems, as requested by Data Translation) and to aid in the identification of reported problems; (d) to install and treat all software releases delivered under this Support Policy as Software in accordance with the terms of the Agreement; and (e) to maintain the Agreement in force and effect.

5. TERM AND TERMINATION.

5.1 Term. For each Product comprising the Software, Support Services will begin on the later of the date the Software warranty granted in the Agreement expires or the date of Licensee's election to obtain Support Services and will apply to such Product for an initial term of one (1) year, unless an alternative commencement date is identified in the Support Order Form. The initial term will automatically be extended for additional terms of one (1) year unless Support Services are terminated at the expiration of the initial term or any additional term, by either party upon thirty (30) days prior written notice to the other party.

5.2 Default. If Licensee is in default of its obligations under the Agreement (except for Licensee's obligation to maintain valid licenses for the Software, in which case termination is immediate) and such default continues for thirty (30) days following receipt of written notice from Data Translation, Data Translation may, in addition to any other remedies it may have, terminate the Support Services.

6. CHARGES, TAXES AND PAYMENTS.

6.1 Payment. The Support Fee in respect of the initial term, and, as adjusted pursuant to Section 5.2 in respect of additional terms, is payable in full prior to the commencement of the initial term or any additional term, as applicable.

6.2 Changes From Term to Term. The Support Fee and the terms and conditions of this Support Policy may be subject to change effective at the end of the initial term or any additional term by giving Licensee at least sixty (60) days prior written notice.

Data Translation Support Policy

6.3 Taxes. The charges specified in this Support Policy are exclusive of taxes. Licensee will pay, or reimburse Data Translation, for all taxes imposed on Licensee or Data Translation arising out of this Support Policy except for any income tax imposed on Data Translation by a governmental entity. Such charges shall be grossed-up for any withholding tax imposed on Data Translation by a foreign governmental entity.

6.4 Additional Charges. Licensee agrees that Data Translation or its authorized distributor will have the right to charge in accordance with Data Translation's then-current policies for any services resulting from (a) Licensee's modification of the Software, (b) Licensee's failure to utilize the then-current release, or the immediately previous Enhanced Release, of the Software, (c) Licensee's failure to maintain Data Translation Support Services throughout the term of the Agreement, (d) problems, errors or inquiries relating to computer hardware or software other than the Software, or (e) problems, errors or inquiries resulting from the misuse or damage or of the Software or from the combination of the Software with other programming or equipment to the extent such combination has not been authorized by Data Translation. Pursuant to Section 2.4 of the Agreement, the Support Fee will also be adjusted in accordance with Data Translation's then current fee schedule as additional Licensed Processors are added. Support Fees do not include travel and living expenses or expenses for installation, training, file conversion costs, optional products and services, directories, shipping charges or the cost of any recommended hardware, third party software, or third party software maintenance fees or operating system upgrade.

7. WARRANTY LIMITATION. EXCEPT AS EXPRESSLY STATED IN THIS SUPPORT POLICY, THERE ARE NO EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO THE SUPPORT SERVICES PROVIDED HEREUNDER (INCLUDING THE FIXING OF ERRORS THAT MAY BE CONTAINED IN THE APPLICABLE DATA TRANSLATION SOFTWARE), INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE WARRANTIES AND REMEDIES SET FORTH IN THIS SUPPORT POLICY ARE EXCLUSIVE, AND ARE IN LIEU OF ALL OTHER WARRANTIES WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED.

8. GENERAL PROVISIONS. Upon the election by Licensee to obtain Support Services, the terms of this Support Policy shall be governed by and are made a part of the Agreement.